Scalable and Distributed Methods for Resolving, Consolidating, Matching and Disambiguating Entities in Linked Data Corpora [1]

Aidan Hogan [a], Antoine Zimmermann [a], Jürgen Umbrich [a], Axel Polleres [a], Stefan Decker [a],

[a] *Digital Enterprise Research Institute, National University of Ireland, Galway*

Abstract

Under review: please do not distribute link or text!
With respect to large-scale, static, Linked Data corpora, in this paper we discuss scalable and distributed methods for: (i) entity consolidation—identifying entities which signify the same referent, aka. smushing, entity resolution, object consolidation, etc.—using explicit `owl:sameAs` relations; (ii) extended entity consolidation based on a subset of OWL 2 RL/RDF rules—particularly over inverse-functional properties, functional-properties and (max-)cardinality restrictions with value one; (iii) deriving weighted concurrence measures between entities in the corpus based on shared inlinks/outlinks and attribute values using statistical analyses; (iv) disambiguating (initially) consolidated entities based on inconsistency detection using OWL 2 RL/RDF rules. Our methods are based upon distributed sorts and scans of the corpus, where we purposefully avoid the requirement for indexing all data. Throughout, we offer evaluation over a diverse Linked Data corpus consisting of 1.118 billion quadruples derived from a domain-agnostic, open crawl of 3.985 million RDF/XML Web documents, demonstrating the feasibility of our methods at that scale, and giving insights into the fecundity of the approach and the quality of the results.

Key words: entity consolidation, web data, linked data, rdf

1. Introduction

Over a decade since the inception of the Semantic Web, RDF publishing has finally found some traction through adoption of Linked Data best practices as follows:

(i) use URIs as names for things (and not just documents);
(ii) make those URIs dereferencable via HTTP;
(iii) return useful and relevant RDF content upon lookup of those URIs;
(iv) include links to other datasets.

The Linked Open Data project has advocated the tangible goal of providing dereferencable machine readable data in a common format (RDF), with emphasis on the re-use of URIs and inter-linkage between remote datasets—in so doing, the project has overseen exports from corporate entities (e.g., the BBC, BestBuy, Freebase), governmental bodies (e.g., the UK Government, the US government), existing structured datasets (e.g., DBPedia), social networking sites (e.g., flickr, Twitter, livejournal), academic communities (e.g., DBLP, UniProt), as well as esoteric exports (e.g., Linked Open Numbers, Poképédia). This burgeoning web of structured data has succinctly been dubbed the "Web of

Email addresses: `aidan.hogan@deri.org` (Aidan Hogan), `antoine.zimmermann@deri.org` (Antoine Zimmermann), `juergen.umbrich@deri.org` (Jürgen Umbrich), `axel.polleres@deri.org` (Axel Polleres), `stefan.decker@deri.org` (Stefan Decker).

[1] The work presented in this paper has been funded in part by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2) and by an IRCSET postgraduate scholarship. Please note that we make our results available for review at `http://swse.deri.org/entity/`.

Data".[2]

Considering the merge of these structured exports, at a conservative estimate there now exists somewhere in the order of tens of billions of RDF triples published on the Web;[3] however, these individual datasets are not well-interlinked[4] (cf. [52])—without sufficient linkage, the ideal of a "Web of Data" quickly disintegrates into the current reality of "Archipelagos of Datasets".

There has been numerous works which have looked at bridging the archipelagos. Some works aim at aligning a small number of related datasets (e.g., [34,42,35]), thus focussing more-so on theoretical considerations than scalability, usually combining symbolic (e.g., reasoning with consistency checking) methods and similarity measures. Some authors have looked at inter-linkage of domain specific RDF datasets at various degrees of scale (e.g., [45,43,27,39,32]). Further research has also looked at exploiting shared terminological data—as well as explicitly asserted links—to better integrate Linked Data collected from thousands or millions of sources (e.g., [22,36,8]); the work presented herein falls most closely into this category. One approach has tackled the problem from the publishers side, detailing a system for manually specifying some (possibly heuristic) criteria for creating links between two datasets [52]. We leave further detailed related work to Section 9.

In this paper, we look at methods to provide better linkage between resources, in particular focussing on finding *equivalent* entities in the data: our notion of an entity is a reference for some individual or *thing* being described by the data—e.g., a person, a place, a musician, a protein, etc; we say that two entities are equivalent if they are coreferent—e.g., refer to the same person, place, etc.[5] Given a collection of datasets which speak about the same referents using different identifiers, we wish to identify these coreferences and somehow merge the knowledge contribution provided by the distinct parties.

In particular, our work is inspired by the requirements of the Semantic Web Search Engine project [24], within which we aim to offer search and browsing over large, static, Linked Data corpora crawled from the Web.[6] The core operation of SWSE is to take user keyword queries as input, and to generate a ranked list of matching entities as results. After the core components of a crawler, index and user-interface, we saw a clear need for a component which *consolidates*—by means of identifying and canonicalising equivalent identifiers—the indexed corpus; there was an observable lack of URI use such that coreferent blank-nodes were prevalent [22] even within the same dataset, and thus we observed many duplicate results referring to the same thing, leading to poor integration of data from our source documents.

In this paper, we revisit this fundamental component, where as we will see in later sections, use and re-use of URIs is still sparse; our core requirements for such a component remain the same:

– the component *must* give **high precision** of consolidated results;
– the underlying algorithm(s) *must* be **scalable**;
– the approach *must* be **fully automatic**;
– the methods *must* be **domain agnostic**;

where a component with poor precision will lead to garbled final results merging unrelated entities, where scalability is required to apply the process over our corpora typically in the order of a billion statements (and which we feasibly hope to expand in future), where the scale of the corpora under analysis precludes any manual intervention, and where—for the purposes of research—the methods should not give preferential treatment to any domain or vocabulary of data (other than core RDF(S)/OWL terms). Alongside these *primary requirements*, we also identify the following *secondary criteria*:

– the analysis *should* demonstrate **high recall**;
– the underlying algorithm(s) *should* be **efficient**;

where the consolidation component should identify as many (correct) equivalences as possible, and where the algorithm should be applicable in reasonable time. Clearly the secondary requirements are also important, but they are superceded by those given earlier, where a certain trade-off exists: we prefer a system which gives a high percentage of correct results and leads to a clean consolidated corpus over an approach which gives a higher percentage of consolidated results but leads to a partially garbled corpus; similarly, we prefer a system

[2] See `http://linkeddata.org/` for Cyganiak and Jentzsch's most current 'map' of the Linked Open Data cloud; however—and as the Linked Open Numbers project (see [53, Figure 1]) has dryly evinced—not all of this current Web of Data is entirely "compelling".

[3] `http://esw.w3.org/TaskForces/CommunityProjects/LinkingOpenData/DataSets/Statistics`

[4] `http://esw.w3.org/TaskForces/CommunityProjects/LinkingOpenData/DataSets/LinkStatistics`

[5] Herein, we avoid philosophical discussion on the notion of identity; for interesting discussion thereon, see [19].

[6] By static, we mean that the system does not cater directly for updates—this omission allows for many optimisations in individual components within the architecture. Instead, we aim at a cyclical indexing paradigm, where the user-facing indices are read-optimised, and the next version of the index is being prepared in background processes.

which can handle more data (is more scalable), but may possibly have a lower throughput (is less efficient). [7]

Thus, in this paper we revisit methods for scalable, precise, automatic and domain-agnostic entity consolidation over large, static Linked Data corpora. In order to make our methods *scalable*, we avoid dynamic on-disk index structures and instead opt for algorithms which rely on sequential on-disk reads/writes of compressed flat files, using operations such as scans, external sorts, merge-joins, and only light-weight or non-critical in-memory indices. In order to make our methods *efficient*, we demonstrate distributed implementations of our methods over a cluster of shared-nothing commodity hardware, where our algorithms attempt to maximise the portion of time spent in embarrassingly parallel execution—i.e., parallel, independent computation without need for inter-machine coordination. In order to make our methods *domain-agnostic* and *fully-automatic*, we exploit the generic formal semantics of the data described in RDF(S)/OWL and also, generic statistics derivable from the corpus. In order to achieve *high recall*, we attempt to exploit—insofar as possible—both the formal semantics and the statistics derivable from the corpus to identify equivalent entities. Aiming at *high precision*, we introduce methods which again exploit the semantics and statistics of the data, but to conversely disambiguate entities—to defeat equivalences found in the previous step which are unlikely to be true according to some criteria.

In particular, in this paper, we:
- provide some necessary preliminaries and describe our distributed architecture (Section 2);
- characterise the 1 billion quadruple Linked Data corpus which will be used for later evaluation of our methods, particular focussing on the (re-)use of data-level identifiers in the corpus (Section 3);
- describe and evaluate our distributed base-line approach for consolidation which leverages explicit `owl:sameAs` relations (Section 4);
- describe and evaluate a distributed approach which extends consolidation to consider a richer OWL semantics for consolidating (Section 5);
- present a distributed algorithm for determining weighted concurrence between entities using statistical analysis of predicates in the corpus (Section 6);
- present a distributed approach to disambiguate entities—i.e., detect likely erroneous consolidation—combining the semantics and statistics derivable from the corpus (Section 7);
- provide critical discussion (Section 8), render related work (Section 9) and conclude with discussion (Section 10).

[7] Of course, entity consolidation has many practical applications outside of our referential use-case SWSE, and is useful in many generic query-answering secnarios—we see these requirements as being somewhat fundamental to a consolidation component, of course to varying degrees.

2. Preliminaries

In this section, we provide some necessary preliminaries relating to (i) RDF: the structured format used in our corpora (Section 2.1); (ii) RDFS/OWL and OWL 2 RL/RDF rules with respect to which we perform *reasoning* and deduce equality (Section 2.2). We attempt to preserve notation and terminology as prevalent in the literature.

2.1. *RDF*

We briefly give some necessary notation relating to RDF constants and RDF triples; see [21].

RDF Constant Given the set of URI references U, the set of blank nodes B, and the set of literals L, the set of *RDF constants* is denoted by $\mathsf{C} = \mathsf{U} \cup \mathsf{B} \cup \mathsf{L}$. As opposed to the RDF-mandated existential semantics for blank-nodes, we interpret blank-nodes as ground skolem constants; note also that we rewrite blank-node labels to ensure uniqueness per document, as prescribed in [21] for merging RDF graphs.

RDF Triple A triple $t := (s, p, o) \in (\mathsf{U} \cup \mathsf{B}) \times \mathsf{U} \times \mathsf{C}$ is called an *RDF triple*, where s is called subject, p predicate, and o object. We call a finite set of triples $G \subset \mathsf{G}$ a *graph*.

RDF Triple in Context/Quadruple Given $c \in \mathsf{U}$, let $\mathtt{http}(c)$ denote the possibly empty graph G_c given by dereferencing the URI c. An ordered pair (t, c) with an RDF triple $t = (s, p, o)$, $c \in \mathsf{U}$ and $t \in \mathtt{http}(c)$ is called a *triple in context c*. We may also refer to (s, p, o, c) as an *RDF quadruple* or quad q with context c.

2.2. *RDFS, OWL, OWL 2 RL/RDF rules and the semantics of equality*

Alongside *assertional data* which defines relationships between *individuals*, provides classes to which those individuals belong, and provide attributes of those individuals which have literal values, RDFS and OWL allow for defining a domain of discourse in the form of *terminological data*, which describes those classes, relationships (object properties), and attributes (datatype properties) and declaratively assigns them a *semantics*. Thereafter, *reasoning* can leverage the semantics of these terms and allows for deriving new knowledge.

OWL 2 RL/RDF [17] rules are a partial-axiomatisation of the OWL 2 RDF-Based Semantics which is applicable for arbitrary RDF graphs, and constitutes an extension of the RDF Semantics [21]. Interestingly for our scenario, this profile includes rules which leverage terminological knowledge to infer equality between individuals—denoted by an `owl:sameAs` relation—and rules which thereafter partially axiomatise the semantics of `owl:sameAs`.

Firstly, in Table B.2, we provide the rules which use terminological knowledge (alongside assertional knowledge) to directly infer `owl:sameAs` relations; we identify rules which require new OWL 2 constructs by italicising the rule label. In Table B.3, we provide an extended set of OWL 2 RL rules which contain precisely one assertional pattern and for which we have demonstrated a scalable implementation in [25]—these rules can provide additional inferences which indirectly lead to the derivation of new `owl:sameAs` data. We will further discuss both sets of rules in Section 5.

In Table B.4, we provide the set of OWL 2 RL/RDF rules which can detect inconsistencies: that is, sets of statements which together form a contradiction—these rules can be used to detect possibly incorrect consolidation, and will be discussed further in Section 7.

In Table B.1, we provide the set of rules which support the (positive) semantics of `owl:sameAs`, axiomatising the reflexivity (**eq-ref**), symmetry (**eq-sym**) and transitivity (**eq-trans**) of the relation, as well as support for the semantics of replacement (**eq-rep-***). Note that we (optionally, and in the case of later evaluation) do not support **eq-ref** or **eq-rep-p**, and provide only partial support for **eq-rep-o**: (i) although **eq-ref** will instead lead to a large bulk of materialised reflexive `owl:sameAs` statements, it is not difficult to see that such statements will not lead to any consolidation or other non-reflexive equality relations; (ii) given that we operate over unverified Web data—and indeed that there is much noise present in such data—we do not want possibly imprecise equality relations to affect predicates of triples, where we support inferencing on such "terminological positions" with alternative reasoning strategies detailed in [25]; (iii) for similar reasons, we do not support replacement for terms in the object position of `rdf:type` triples.

Given that the semantics of equality is quadratic with respect to the A-Box, we apply a partial-materialisation approach which gives our notion of *consolidation*—instead of materialising all possible inferences given by the semantics of replacement, we instead choose one "canon" to represent the set of equivalent terms. We have used this approach in previous works [22–24], and it has also appeared in related works in the literature [51,28], as a common-sense optimisation for handling data-level equality. To take an example, in previous works [22] we found a valid equivalence class (set of equivalent entities) with 32,390 members; materialising all non-reflexive `owl:sameAs` statements would infer more than 1 billion `owl:sameAs` relations($32{,}390^2$ - 32,390) = 1,049,079,710; further assuming that each entity appeared in, on average, two quadruples, we would infer an additional $\sim$2 billion of massively duplicated data.

Note that although we only perform partial materialisation—and with the exception of not supporting **eq-ref-p** and only partially supporting **eq-rep-o**—we do not change the semantics of equality: alongside the partially materialised data, we provide a set of consolidated `owl:sameAs` relations (containing all of the identifiers in each equivalence class) which can be used to "backward-chain" the full inferences possible through replacement (as required).[8] Thus, we do not consider the canon as somehow 'definitive' or superceding the other identifiers, but merely consider it as *representing* the equivalence class.[9]

Finally, herein we do not consider consolidation of literals; one may consider useful applications, e.g., for canonicalising datatype literals, but such discussion is out of the current scope.

2.3. *Distribution architecture*

Our methods are implemented on a shared-nothing distributed architecture [47] over a cluster of commodity hardware. The distributed framework consists of a master machine which orchestrates the given tasks, and several slave machines which perform parts of the task in parallel.

The master machine can instigate the following distributed operations:

[8] With respect to **eq-ref**, one can consider fairly trivial backward-chaining (or query-time) support for said semantics.

[9] We may optionally consider non-canonical blank-node identifiers as redundant and discard them.

- **scatter:** partition on-disk data using some local *split* function, and send each chunk to individual slave machines for subsequent processing;
- **run:** request the parallel execution of a task by the slave machines—such a task either involves processing of some data local to the slave machine, or the **coordinate** method (described later) for reorganising the data under analysis;
- **gather:** gathers chunks of output data from the slave swarm and performs some local *merge* function over the data;
- **flood:** broadcast global knowledge required by all slave machines for a future task.

The master machine provides input data to the slave swarm, provides the control logic required by the distributed task (commencing tasks, coordinating timing, ending tasks), gathers and locally perform tasks on global knowledge which the slave machines would otherwise have to replicate in parallel, and transmits globally required knowledge.

The slave machines, as well as performing tasks in parallel, can perform the following distributed operation (on the behest of the master machine):

- **coordinate:** local data on each slave machine is partitioned according to some *split* function, with the chunks sent to individual machines in parallel; each slave machine also gathers the incoming chunks in parallel using some *merge* function.

The above operation allows slave machines to reorganise (split/send/gather) intermediary amongst themselves; the **coordinate** operation could be replaced by a pair of **gather**/**scatter** operations performed by the master machine, but we wish to avoid the channelling of all intermediary data through one machine. Without the **coordinate** operation, our framework closely resembles the MapReduce framework [11], with **scatter** corresponding to the *Map* operation, and **gather** corresponding to the *Reduce* operation.

Note that herein, we assume that the input corpus is evenly distributed and split across the slave machines, and that the slave machines have roughly even specifications: that is, we do not consider any special form of load balancing, but instead aim to have uniform machines processing comparable data-chunks.

We note that there is the practical issue of the master machine being idle waiting for the slaves, and, more critically, the potentially large cluster of slave machines waiting idle for the master machine. One could overcome idle times with mature task-scheduling (e.g., interleaving jobs) and load-balancing. From an algorithmic point of view, removing the central coordination on the master machine may enable better distributability. One possibility would be to allow the slave machines to duplicate the aggregation of global knowledge in parallel: although this would free up the master machine and would probably take roughly the same time, duplicating computation wastes resources which could otherwise be exploited by, e.g., interleaving jobs. A second possibility would be to avoid the requirement for global knowledge and to coordinate upon the larger corpus (e.g., a **coordinate** function hashing on the subject and object of the data, or perhaps an adaptation of the SpeedDate routing strategy [37]). Such decisions are heavily influenced by the scale of the task to perform, the percentage of knowledge which is globally required, how the input data are distributed, how the output data should be distributed, and the nature of the cluster over which it should be performed and the task-scheduling possible. The distributed implementation of our tasks are designed to exploit a relatively small percentage of global knowledge which is cheap to coordinate, and we choose to avoid—insofar as reasonable—duplicating computation.

2.4. *Experimental setup*

Our entire code-base is implemented on top of standard Java libraries; we thus instantiate the distributed architecture using Java RMI libraries, and using the lightweight open-source Java RMIIO package[10] for streaming data for the network.

All of our evaluation is based on nine machines connected by Gigabit ethernet,[11] each with uniform specifications, viz., 2.2GHz Opteron x86-64, 4GB main memory, 160GB SATA hard-disks, running Java 1.6.0_12 on Debian 5.0.4.

3. Experimental corpus

Later in this paper, we discuss the performance and fecundity of applying our methods over a corpus of 1.118 billion quadruples derived from an RDF/XML crawl of 3.985 million web documents in mid-May 2010 (detailed in [24]. Of the 1.118 billion quads, 1.106 billion are unique, and 947 million are unique triples. The data contain 23 thousand unique predicates and 105 thousand unique class terms (terms in the object position of an `rdf:type` triple). In terms of diversity, the corpus consists of data from 783 pay-level-domains—

[10] `http://openhms.sourceforge.net/rmiio/`
[11] We observe, e.g., a max FTP transfer rate of 38MB/sec between machines.

PLDs; e,g, `deri.ie`, `data.gov.uk`—as was the result of an open, domain agnostic crawl [24].

Now, we discuss the usage of terms in a data-level position, viz., terms in the subject position or object position of non-`rdf:type` triples.[12] Since we do not consider the consolidation of literals or schema-level concepts, we focus on characterising blank-node and URI re-use in such data-level positions, thus rendering a picture of the morphology of the data.

We found 286.3 million unique terms, of which 165.4 million (57.8%) were blank-nodes, 92.1 million (32.2%) were URIs, and 28.9 million (10%) were literals. With respect to literals, each had on average 9.473 data-level occurrences (by definition, all in the object position).

With respect to blank-nodes, each had on average 5.233 data-level occurrences. Each occurred on average 0.995 times in the object position of a non-`rdf:type` triple, with 3.1 million (1.87%) not occurring in the object position; conversely, each occurred on average 4.239 times in the subject position of a triple, with 69 thousand (0.04%) not occurring in the subject position. Thus, we summarise that almost all blank-nodes appear in both the subject position and object position, but occur most prevalently in the former. Importantly, note that in our input, blank-nodes cannot be re-used across sources.

With respect to URIs, each had on average 9.41 data-level occurrences (1.8$\times$ the average for blank-nodes), with 4.399 average appearances in the subject position and 5.01 appearances in the object position—19,85 million (21.55%) did not appear in an object position, whilst 57.91 million (62.88%) did not appear in a subject position.

With respect to re-use across sources, each URI had a data-level occurrence in, on average, 4.7 documents, and 1.008 PLDs—56.2 million (61.02%) of URIs appeared in only one document, and 91.3 million (99.13%) only appeared in one PLD. Also, re-use of URIs across documents was heavily weighted in favour of use in the object position: URIs appeared in the subject position in, on average, 1.061 documents and 0.346 PLDs; for the object position of non-`rdf:type` triples, URIs occurred in, on average, 3.996 documents and 0.727 PLDs.

The URI with the most data-level occurrences (1.66 million) was `http://identi.ca/`; the URI with the most re-use across documents (appearing in 179.3 thousand documents) was `http://creativecommons.org/licenses/by/3.0/`; the URI with the most re-use across PLDs (appearing in 80 different domains) was `http://www.ldodds.com/foaf/foaf-a-matic`. Although some URIs do enjoy widespread re-use across different documents and domains, in Figures 1 and 2 we give the distribution of re-use of URIs across documents and across PLDs, where a power-law relationship is roughly evident—again, the majority of URIs only appear in one document (61%) or in one PLD (99%).

[12]Please see [24, Appendix A] for further statistics relating to this corpus.

From this analysis, we can conclude that with respect to data-level terms in our corpus:

– blank-nodes—which by their very nature cannot be re-used across documents—are 1.8$\times$ more prevalent than URIs;
– despite a smaller number of unique URIs, each one is used in (probably coincidentally) 1.8$\times$ more triples;
– unlike blank-nodes, URIs commonly only appear in either a subject position or an object position;
– each URI is re-used on average in 4.7 documents, but usually only within the same domain—most external re-use is in the object position of a triple;
– 99% of URIs appear in only one PLD.

We can conclude that within our corpus—itself a general crawl for RDF/XML on the Web—we find that there is only sparse re-use of data-level terms across sources and particularly domains.

4. Base-line Consolidation

We now present the “base-line” algorithm for consolidation which leverages asserted `owl:sameAs` relations in the data—this is the current method of consolidation employed by the SWSE system and has been described in [24].[13] Herein, we briefly reintroduce the approach and performance details and then provide extended analysis of the results.

4.1. *High-level approach*

The approach is straight-forward:

(i) scan the corpus and separate out all asserted `owl:sameAs` relations from the main body of the corpus;

[13]**Note to reviewer:** an earlier version of a SWSE paper was submitted to the JWS Special Issue on Semantic Search in February and we are still awaiting notification—for reference, please see an updated technical report version of the paper which contains the experiments over the same corpus [24]—and in this previous paper, we describe our baseline approach as herein reintroduced. This should not affect novelty since (i) the SWSE paper covers the end-to-end system of which consolidation is only one component; (ii) in this paper, the base-line approach is heavily extended upon in later sections.

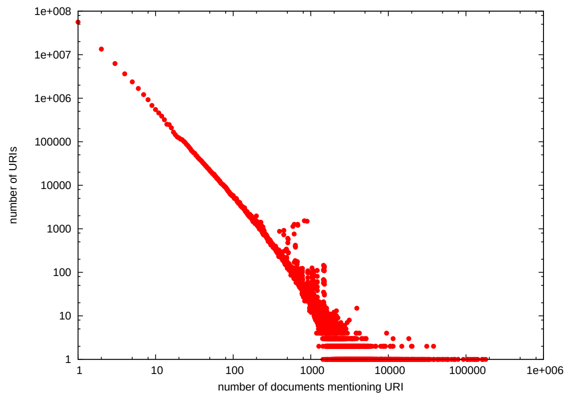


Fig. 1. Distribution of URIs and the number of documents they appear in (in a data-position)

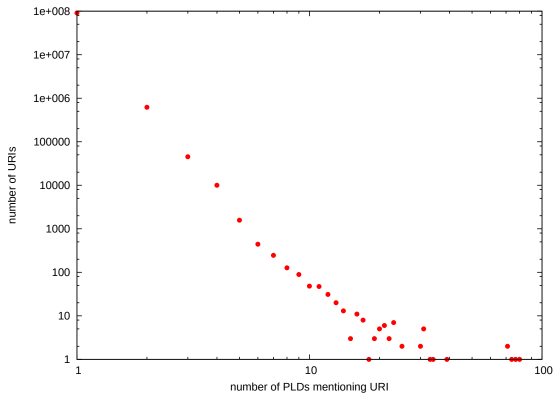


Fig. 2. Distribution of URIs and the number of PLDs they appear in (in a data-position)

(ii) load these relations into an in-memory index, which encodes the transitive and symmetric semantics of `owl:sameAs`;
(iii) for each equivalence class in the index, choose a canonical term;
(iv) scan the corpus again, canonicalising any term in the subject position or object position of an `rdf:type` triple.

Thus, we need only index a small subset of the corpus—`owl:sameAs` statements—and can apply consolidation by means of two scans. The non-trivial aspects of the algorithm are given by the in-memory equality index: we provide the details in Algorithm 1, where we use a map which stores a term (involved in a non-reflexive equality relation) as key, and stores a flat set of equivalent terms (of which the key is a member) as value—thus, we can perform a lookup of any term and retrieve the set of equivalent terms given by the `owl:sameAs` corpus.

With respect to choosing a canonical term, we prefer URIs over blank-nodes, thereafter choosing a term with the lowest alphabetical ordering; a canonical term is then associated to each equivalent set. Once the equivalence index has been finalised, we re-scan the corpus and canonicalise the data.

4.2. *Distributed approach*

Again, distribution of the approach is fairly intuitive, as follows:

(i) **run**: scan the distributed corpus (split over the slave machines) in parallel to extract `owl:sameAs` relations;
(ii) **gather**: gather all `owl:sameAs` relations onto the master machine, and build the in-memory equality index;

Algorithm 1 Building equivalence map (from [24])

```
Require: SAMEAS DATA : $\mathcal{SA}$
1:  map ← {}
2:  for $t \in \mathcal{SA}$ do
3:      $eqc_s \leftarrow$ map.get($t.s$)
4:      if $eqc_s = \emptyset$ then
5:          $eqc_s \leftarrow \{s\}$
6:      end if
7:      $eqc_o \leftarrow$ map.get($t.o$)
8:      if $eqc_o = \emptyset$ then
9:          $eqc_o \leftarrow \{o\}$
10:     end if
11:     if $eqc_s \neq eqc_o$ then
12:         $eqc_{s\cup o} \leftarrow eqc_s \cup eqc_o$
13:         for $e \in eqc_{s\cup o}$ do
14:             map.put($e, eqc_{s\cup o}$)
15:         end for
16:     end if
17: end for
```

(iii) **flood/run**: send the equality index (in its entirety) to each slave machine, and apply the consolidation scan in parallel.

As we will see in the next section, the most expensive methods—involving the two scans of the main corpus—can be conducted in parallel.

4.3. *Performance Evaluation*

We applied the distributed base-line consolidation over our corpus with the aformentioned procedure and setup. The entire consolidation process took 63.3 min, with the bulk of time taken as follows: the first scan extracting `owl:sameAs` statements took 12.5 min, with an average idle time for the servers of 11 s (1.4%)—i.e., on average, the slave machines spent 1.4% of the time

Category	**min**	**% Total**
Total execution time	63.3	100
Master (Local)		
Executing	8.5	13.4
Aggregating `owl:sameAs`	8.4	13.3
Miscellaneous	0.1	0.1
Idle (waiting for slaves)	54.8	86.6
Slave (Parallel)		
Avg. Executing (total)	53.5	84.6
Extract `owl:sameAs`	12.3	19.5
Consolidate	41.2	65.1
Avg. Idle	9.8	15.4
Waiting for peers	1.3	2
Waiting for master	8.5	13.4

Table 1
Breakdown of timing of distributed baseline consolidation

idly waiting for peers to finish. Transferring, aggregating and loading the `owl:sameAs` statements on the master machine took 8.4 min. The second scan rewriting the data according to the canonical identifiers took in total 42.3 min, with an average idle time of 64.7 s (2.5%) for each machine at the end of the round. The slower time for the second round is attributable to the extra overhead of re-writing the data to disk, as opposed to just reading.

In Table 1, we give a breakdown of the timing for the tasks. Of course, please note that the percentages are a function of the number of machines where, e.g., a higher number of slave machines will correspond to a higher percentage of time on the master machine. However, independent of the number of slaves, we note that the master machine required 8.5 min for coordinating globally-required knowledge `owl:sameAs`, and that the rest of the task time is spent in embarrassingly parallel execution (amenable to reduction by increasing the number of machines). For our setup, the slave machines were kept busy for, on average, 84.6% of the total task time; of the idle time, 87% was spent waiting for the master to coordinate the `owl:sameAs` data, and 13% was spent waiting for peers to finish their task due to sub-optimal load balancing. The master machine spent 86.6% of the task idle waiting for the slaves to finish.

4.4. *Results Evaluation*

We extracted 11.93 million raw `owl:sameAs` statements, forming 2.16 million equivalence classes mentioning 5.75 million terms (6.24% of URIs)—an average of 2.65 elements per equivalence class. Of the 5.75 million terms, only 4,156 were blank-nodes. Figure 3 presents the distribution of sizes of the equivalence classes, where the largest equivalence class contains 8,481 equivalent entities and 1.6 million (74.1%) equivalence classes contain the minimum two equivalent identifiers.

#	**PLD**	**PLD**	**Co-occur**
1	`rdfize.com`	`uriburner.com`	506,623
2	`dbpedia.org`	`freebase.com`	187,054
3	`bio2rdf.org`	`purl.org`	185,392
4	`loc.gov`	`info:lc/authorities`[a]	166,650
5	`l3s.de`	`rkbexplorer.com`	125,842
6	`bibsonomy.org`	`l3s.de`	125,832
7	`bibsonomy.org`	`rkbexplorer.com`	125,830
8	`dbpedia.org`	`mpii.de`	99,827
9	`freebase.com`	`mpii.de`	89,610
10	`dbpedia.org`	`umbel.org`	65,565

Table 3
Top 10 PLD pairs co-occurring in the equivalence classes, with number of equivalence classes they co-occur in

[a] In fact, these were 'syntactic' equivalences within the same PLD

Table 2 shows the canonical URIs for the largest 5 equivalence classes; we manually inspected the results and show whether or not the results were verified as correct/incorrect. Indeed, results for class **1** and **2** were deemed incorrect due to over-use of `owl:sameAs` for linking drug-related entities in the DailyMed and LinkedCT exporters. Results **3** and **5** were verified as correct consolidation of prominent Semantic Web related authors, resp.: `Dieter Fensel` and `Rudi Studer`—authors are given many duplicate URIs by the RKBExplorer coreference index.[14] Result **4** contained URIs from various sites generally referring to the United States, mostly from DBPedia and LastFM. With respect to the DBPedia URIs, these (i) were equivalent but for capitilisation variations or stop-words, (ii) were variations of abbreviations or valid synonyms, (iii) were different language versions (e.g., `dbpedia:États_Unis`), (iv) were nicknames (e.g., `dbpedia:Yankee_land`), (v) were related but not equivalent (e.g., `dbpedia:American_Civilization`), (vi) were just noise (e.g., `dbpedia:LOL_Dean`).

Besides the largest equivalence classes—which we have seen are prone to errors perhaps due to the snowballing effect of the transitive and symmetric closure—we also randomly sampled 100 equivalent sets and manually checked for errors based on label (as an intuitive idea of what the identifier refers to) and type. We veri-

[14]For example, see the coreference results given by `http://www.rkbexplorer.com/sameAs/?uri=http://acm.rkbexplorer.com/id/person-53292-22877d02973d0d01e8f29c7113776e7e`, which at the time of writing correspond to 436 out of the 443 equivalent URIs found for `Dieter Fensel`.

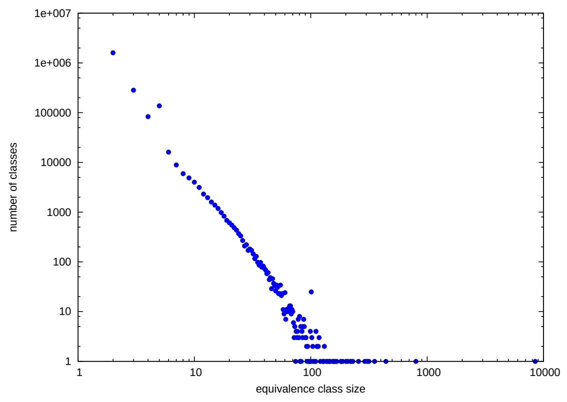


Fig. 3. Distribution of sizes of equivalence classes on log/log scale (from [24])

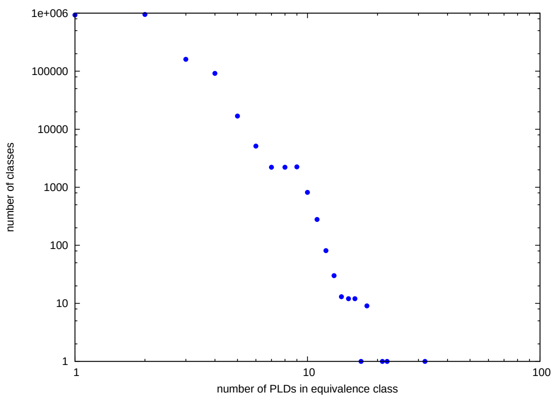


Fig. 4. Distribution of the number of PLDs per equivalence class on log/log scale

#	Canonical Term (Lexically Lowest in Equivalence Class)	Size	Correct?
1	`http://bio2rdf.org/dailymed_drugs:1000`	8,481	×
2	`http://bio2rdf.org/dailymed_drugs:1042`	800	×
3	`http://acm.rkbexplorer.com/id/person-53292-22877d02973d0d01e8f29c7113776e7e`	443	✓
4	`http://agame2teach.com/#ddb61cae0e083f705f65944cc3bb3968ce3f3ab59-ge_1`	353	✓/×
5	`http://acm.rkbexplorer.com/id/person-236166-1b4ef5fdf4a5216256064c45a8923bc9`	316	✓

Table 2
Largest 5 equivalence classes (from [24])

fied that all 100 were correct (or, more accurately, were not obviously incorrect).[15]

In Table 3, we give the most frequently co-occurring PLD-pairs in our equivalence classes, where datasets resident on these domains are "heavily" interlinked with `owl:sameAs` relations.

With respect to consolidation, identifiers in 78.6 million subject positions (7% of subject positions) and 23.2 million non-`rdf:type`-object positions (2.6%) were rewritten, giving a total of 101.9 million positions rewritten (5.1% of total rewritable positions). The average number of documents mentioning each URI rose slightly from 4.691 to 4.719 (a 0.6% increase) due to consolidation, and the average number of PLDs also rose slightly from 1.005 to 1.007 (a 0.2% increase).

5. Extended Reasoning Consolidation

Having presented the baseline approach as currently used in SWSE [24], we look at extending the approach to include more expressive reasoning capabilities, using OWL 2 RL/RDF rules to infer novel `owl:sameAs` relations.

[15]Many were simple 'syntactic' equivalences from the `opiumfield.com` LastFM data exporter; for reference, we've published the 100 sets at `http://aidanhogan.com/swse/eqcs-sample-100.txt`.

5.1. *High-level approach*

In Table B.2, we provide the pertinent rules for inferring new `owl:sameAs` relations from the data. However, after analysis of the data, we observed that no documents used the `owl:maxQualifiedCardinality` construct required for the **cls-maxqc*** rules, and that only one document defined one `owl:hasKey` axiom[16] involving properties with < 5 occurrences in the data—hence, we leave implementation of these rules for future work and note that these new OWL 2 constructs have probably not yet had time to find proper traction on the Web. Thus, on top of inferencing involving explicit `owl:sameAs`, we are left with **prp-fp** which supports the semantics of properties typed `owl:FunctionalProperty`, **prp-ifp** which supports the semantics of properties typed `owl:InverseFunctionalProperty`, and **cls-maxc1** which supports the semantics of classes with a specified cardinality of `1` for some defined property (a class restricted version of the functional-property inferencing). Note that we have presented non-distributed execution of these rules at a smaller scale (147 million quadruples) in [23].

Thus, we look at using OWL 2 RL/RDF rules **prp-fp**, **prp-ifp** and **cls-maxc2** for inferring new `owl:sameAs` relations between individuals—we also support an ad-

[16]`http://huemer.lstadler.net/role/rh.rdf`

ditional rule which gives an exact cardinality version of **cls-maxc2**.[17]

However, applying only these rules may lead to incomplete results; for example, consider the following data:

```
# From the FOAF Vocabulary
 foaf:homepage rdfs:subPropertyOf foaf:isPrimaryTopicOf .
 foaf:isPrimaryTopicOf owl:inverseOf foaf:primaryTopic .
 foaf:isPrimaryTopicOf a owl:InverseFunctionalProperty .
# From DBLP publications export
 dblp:Axel foaf:homepage <http://polleres.net/> .
# From Axel's FOAF file
 <http://polleres.net/> foaf:primaryTopic axel:me .
```

Here we need OWL 2 RL/RDF rules **prp-inv** and **prp-spo1**—handling standard `owl:inverseOf` and `rdfs:subPropertyOf` inferencing respectively—to infer:

```
# From prp-spo1
 dblp:Axel foaf:isPrimaryTopicOf <http://polleres.net/> .
# From prp-inv
 axel:me foaf:isPrimaryTopicOf <http://polleres.net/> .
# Subsequently, from prp-ifp
 dblp:Axel owl:sameAs axel:me .
 axel:me owl:sameAs dblp:Axel .
```

Thus, we also investigate pre-applying more general OWL 2 RL/RDF reasoning over the corpus to derive more complete results, where the ruleset is available in Table B.3 and is restricted to OWL 2 RL/RDF rules with one assertional pattern. Herein, we only sketch the details of applying these "general rules"—for which we use the distributed Scalable Authoritative OWL Reasoner (SAOR) detailed in [25]—focusing instead upon the application of rules which directly produce `owl:sameAs` consequences.

Thus, our high-level approach is as follows:

(i) extract relevant terminological data from the corpus;
(ii) bind the terminological patterns in the rules from this data, thus creating a larger set of general rules with only one assertional pattern and identifying assertional patterns which are useful for consolidation;
(iii) apply general-rules over the corpus, and buffer any input/inferred statements relevant for consolidation to a new file;
(iv) derive the closure of `owl:sameAs` statements from the consolidation-relevant dataset;
(v) apply consolidation over the main corpus with respect to the closed `owl:sameAs` data.

In Step (i), we extract terminological data required for application of both the general/consolidation rules, including members of the classes `owl:(Inverse)FunctionalProperty` and relevant cardinality restrictions. During the extraction and analysis of the terminological data, we only consider statements which are authoritatively served by a document; details of the authoritative analysis are available in [25], but to give a brief anecdotal explanation, we only consider—for example—an axiom stating that a property `P` is inverse-functional if served by the document to which `P` deferences: i.e., we would only consider such axioms on FOAF properties if given by the FOAF spec, and ignore third-party axioms saying that, e.g.,

```
foaf:name a owl:InverseFunctionalProperty .
foaf:weblog a owl:FunctionalProperty .
```

In Step (ii), we (authoritatively) bind the terminological patterns of the general rules and the join variables in the assertional patterns to create a set of rules with only one assertional pattern, amenable to execution by a single scan; e.g.:

```
   ?x foaf:homepage ?y .
⇒ ?x isPrimaryTopicOf ?y .
```

Again, the process is detailed in [25]. We also bind the terminological patterns of the consolidation rules in Table B.2 to identify patterns, such as:

```
?x foaf:isPrimaryTopicOf ?y .
```

which are useful for consolidation.

In Step (iii), we apply the (terminologically ground) general rules over the corpus, where any input or inferred statements matching a consolidation relevant pattern are buffered to a separate file, including any `owl:sameAs` statements found.

Subsequently, in Step (iv), we must now compute the *canonicalised closure* of the `owl:sameAs` statements. In the previous section we used an in-memory equality index to support the semantics of `owl:sameAs`, to represent the equivalence classes and chosen canonical terms, and to provide the lookups required during canonicalisation of the corpus. However, by using such an approach, the scalability of the system is bound by the memory resources of the hardware (which itself cannot be solved by distribution since—in our approach—all machines require knowledge about all same-as statements). In particular, the extended reasoning approach will produce a large set of such statements which will require a prohibitive amount of memory to store.[18] Thus, we turn to

[17]Exact cardinalities are disallowed in OWL 2 RL due to their effect on the formal proposition of completeness underlying the profile, but such considerations are moot in our scenario.

[18]Currently, we store entire (uncompressed) strings in memory, using a flyweight pattern (interning) which guarantees unique references. One could consider lossless string compression techniques over the repetitive URI strings (e.g., see [31,14]) to increase the in-memory

on-disk methods to handle the transitive and symmetric closure of the `owl:sameAs` corpus and to perform the subsequent consolidation of the corpus in Step (v).

In particular, we mainly employ the following three on-disk primitives: (i) **sequential scans** of flat files containing line-delimited tuples;[19] (ii) **external-sorts** where batches of statements are sorted in memory, the sorted batches written to disk, and the sorted batches merged to the final output; and (iii) **merge-joins** where multiple sets of data are sorted according to their required join position, and subsequently scanned in an interleaving manner which aligns on the join position and where an in-memory join is applied for each individual join element. Using these primitives to perform the `owl:sameAs` computation minimises the amount of main memory required, where we have presented similar approaches in [22,23].

First, assertional memberships of functional properties, inverse-functional properties and cardinality restrictions (both properties and classes) are written to separate on-disk files. For functional-property and cardinality reasoning, a consistent join variable for the assertional patterns is given by the subject position; for inverse-functional-property reasoning, a join variable is given by the object position.[20] Thus, we can sort the former sets of data according to subject and perform a merge-join by means of a linear scan thereafter; the same procedure applies to the latter set, sorting and merge-joining on the object position. Applying merge-join scans, we produce new `owl:sameAs` statements.

Both the originally asserted and newly inferred `owl:sameAs` relations are similarly written to an on-disk file, over which we now wish to perform the canonicalised symmetric/transitive closure. We apply a similar method again, leveraging external sorts and merge-joins to perform the computation (herein, we sketch and point the interested reader to [23, Section 4.6]). In the following, we use $>$ and $<$ to denote lexical ordering, `SA` as a shortcut for `owl:sameAs`, a, b, c, etc., to denote members of $\mathsf{U} \cup \mathsf{B}$ such that $a < b < c$, and define URIs to be lexically lower than blank nodes ($\forall u \in \mathsf{U}, \forall v \in \mathsf{B} : u < v$). The process is as follows:

(i) we only materialise symmetric equality relations which involve a (possibly intermediary) canonical term chosen by a lexical ordering: given b `SA` a, we materialise a `SA` b; given a `SA` b `SA` c, we materialise the relations a `SA` b, a `SA` c, and their inverses, but do not materialise b `SA` c or its inverse;

(ii) transitivity is supported by iterative merge-join scans:
  - during the scan, if we find c `SA` a (sorted by object) and c `SA` d (sorted naturally), we infer a `SA` d and drop the non-canonical c `SA` d (and d `SA` c);
  - at the end of the scan:
    - · newly inferred triples are marked and merge-joined into the main body of equality relations—any triples echoing an earlier inference are ignored;
    - · dropped non-canonical statements are removed;
  - the process is then iterative: in the next scan, if we find d `SA` a and d `SA` e, we infer a `SA` e and e `SA` a;
  - inferences will only occur if they involve a statement added in the previous iteration, ensuring that inference steps are not re-computed and that the computation will terminate;

(iii) the above iterations stop when a fixpoint is reached and nothing new is inferred;

(iv) the process of reaching a fixpoint is accelerated using available main-memory to store a cache of partial equality chains.

With respect to the last item, we use Algorithm 1 to derive "batches" of in-memory equivalences, and when in-memory capacity is achieved, we write these batches to disk and proceed with on-disk computation: this is particularly useful for computing the small number of long equality chains which would otherwise require sorts and merge-joins over all of the canonical `owl:sameAs` data currently derived, and where the number of iterations would otherwise be the length of the longest chain.

The result of this process is a set of canonicalised equality relations representing the symmetric/transitive closure. Finally, we briefly describe the process of canonicalising data with respect to this on-disk equality corpus, where we again use external-sorts and merge-joins. Firstly, we prune the `owl:sameAs` index to only maintain relations s_1 `SA` s_2 such that $s_1 > s_2$—thus, given s_1 `SA` s_2, we know that s_2 is the canon, and s_1 is to be rewritten. We then sort the data according to the position which we wish to rewrite, and perform a merge-join over both sets of data, buffering the canonicalised data to an output file. If we want to rewrite mul-

capacity. We are reluctant to adopt an OID approach, viewing the OID indexing and mapping from OIDs to full strings as prohibitively expensive, subject to further investigation.

[19]These files are G-Zip compressed flat files of N-Triple-like syntax encoding arbitrary length tuples of RDF constants.

[20]Although a predicate-position join is also available, we prefer data-position joins which provide smaller batches of data for the in-memory join.

tiple positions of a file of tuples (e.g., subject and object), we must rewrite one position, sort the results by the second position, and subsequently rewrite the second position. [21]

Finally, note that in the derivation of `owl:sameAs` from the consolidation rules **prp-fp**, **prp-ifp**, **cax-maxc2**, the overall process may be iterative. For instance, consider the following data:

```
dblp:Axel foaf:isPrimaryTopicOf <http://polleres.net/> .
axel:me foaf:isPrimaryTopicOf <http://axel.deri.ie/> .
<http://polleres.net/> owl:sameAs <http://axel.deri.ie/> .
```

from which the conclusion that `dblp:Axel` is the same as `axel:me` holds. We see that new `owl:sameAs` relations (either asserted or derived from the consolidation rules) may in turn "align" terms in the join position of the consolidation rules, leading to new equivalences. Thus, for deriving the final `owl:sameAs`, we require a higher-level iterative process as follows:

(i) initially apply the consolidation rules, and append the results to a file alongside the asserted `owl:sameAs` statements found;
(ii) apply the initial closure of the `owl:sameAs` data;
(iii) then, iteratively until no new `owl:sameAs` inferences are found:
  – rewrite the join positions of the on-disk files containing the data for each consolidation rule according to the current `owl:sameAs` data;
  – derive new `owl:sameAs` inferences possible through the previous rewriting for each consolidation rule;
  – re-derive the closure of the `owl:sameAs` data including the new inferences.

The final closed file of `owl:sameAs` data can then be re-used to rewrite the main corpus in two sorts and merge-join scans over subject and object.

5.2. *Distributed approach*

The distributed approach follows quite naturally from the previous discussion. As before, we assume that the input data are evenly pre-distributed over the slave machines (in any arbitrary ordering), where we can then apply the following process:

(i) **run**: scan the distributed corpus (split over the slave machines) in parallel to extract relevant terminological knowledge;
(ii) **gather**: gather terminological data onto the master machine and thereafter bind the terminological patterns of the general/consolidation rules;
(iii) **flood**: flood the rules for reasoning and the consolidation-relevant patterns to all slave machines;
(iv) **run**: apply reasoning and extract consolidation-relevant statements from the input and inferred data;
(v) **gather**: gather all consolidation statements onto the master machine, then in parallel:
  – **local**: compute the closure of the consolidation rules and the `owl:sameAs` data on the master machine;
  – **run**: each slave machine sorts its fragment of the main corpus according to natural order (s, p, o, c);
(vi) **flood**: send the closed `owl:sameAs` data to the slave machines once the distributed sort has been completed;
(vii) **run**: each slave machine then rewrites the subjects of their segment of the corpus, subsequently sorts the rewritten data by object, and then rewrites the objects (of non-`rdf:type` triples) with respect to the closed `owl:sameAs` data.

5.3. *Performance Evaluation*

Applying the above process to our 1.118 billion quadruple corpus took 12.34 h. Extracting the terminological data took 1.14 h with an average idle time of 19 min (27.7%) (one machine took ∼18 min longer than the rest due to processing a large ontology containing ∼2 million quadruples [24]). Merging and aggregating the terminological data took roughly ∼1 min. Applying the reasoning and extracting the consolidation relevant statements took 2.34 h, with an average idle time of 2.5 min (1.8%). Aggregating and merging the consolidation relevant statements took 29.9 min. Thereafter, locally computing the closure of the consolidation rules and the equality data took 3.52 h, with the computation requiring two iterations overall (the minimum possible—the second iteration did not produce any new results); concurrent to the previous step, the parallel sort of remote data by natural order took 2.33 h with an average idle time of 6 min (4.3%). Subsequent parallel consolida-

[21] One could consider instead building an on-disk map for equivalence classes and pivot elements and follow a consolidation procedure similar to the previous section over the unordered corpus: however, we would expect that such an on-disk index would have a low cache hit-rate given the nature of the data, which would lead to a high number of disk seek operations. An alternative approach might be to split and hash the corpus according to subject/object and split the equality data into relevant segments loadable in-memory on each machine: however, this would again require a non-trivial minimum amount of memory to be available over the given cluster.

Category	**min**	**% Total**
Total execution time	740.4	100
Master (Local)		
Executing	243.6	32.9
Aggregate Consolidation Relevant Data	29.9	4
Closing `owl:sameAs`†	211.2	28.5
Miscellaneous	2.5	0.3
Idle (waiting for slaves)	496.8	67.1
Slave (Parallel)		
Avg. Executing (total)	599.1	80.9
Extract Terminology	49.4	6.7
Extract Consolidation Relevant Data	137.9	18.6
Initial Sort (by subject)†	133.8	18.1
Consolidation	278	37.5
Avg. Idle	141.3	19.1
Waiting for peers	37.5	5.1
Waiting for master	103.8	14

Table 4
Breakdown of timing of distributed extended consolidation w/reasoning where † identifies the master/slave tasks run concurrently

tion of the data took 4.8 h with 10 min (3.5%) average idle time—of this, ∼19% of the time was spent consolidating the pre-sorted subjects, ∼60% of the time was spent sorting the rewritten data by object, and ∼21% of the time was spent consolidating the objects of the data.

As before, Table 4 summarises the timing of the task, where the master machine requires 4.06 h to coordinate global knowledge, constituting the lower bound on time possible for the task to execute with respect to increasing machines in our setup—in future it may be worthwhile to investigate distributed strategies for computing the `owl:sameAs` closure (which takes 28.5% of the total computation time), but for the moment we mitigate the cost by concurrently running a sort on the slave machines, thus keeping the slaves busy for 63.4% of the time taken for this local aggregation step. The slave machines were, on average, busy for 80.9% of the total task time; of the idle time, 73.3% was spent waiting for the master machine to aggregate the consolidation relevant data and to finish the closure of `owl:sameAs` data, and the balance (26.7%) was spent waiting for peers to finish (mostly during the extraction of terminological data).

Briefly, we also ran the consolidation without the general reasoning rules (Table B.3) motivated earlier. With respect to performance, the main variations were given by (i) the extraction of consolidation relevant statements—this time directly extracted from explicit statements as opposed to explicit and inferred statements—which took 15.4 min (11% of the time taken including the general reasoning) with an average idle time of less than one minute (6% average idle time); (ii) local aggregation of the consolidation relevant statements took 17 min (56.9% of the time taken previously); (iii) local closure of the `owl:sameAs` data took 3.18 h (90.4% of the time taken previously). The total time saved equated to 2.8 h (22.7%), where 33.3 min were saved from coordination on the master machine, and 2.25 h were saved from parallel execution on the slave machines.

5.4. *Results Evaluation*

Note that in this section, we present the results of the consolidation which included the general reasoning step in the extraction of consolidation-relevant statements. In fact, we found that the only major variation between the two approaches was in the amount of consolidation-relevant statements collected (discussed presently), where other variations were in fact negligible (<0.1%). Thus, for our corpus, extracting only asserted consolidation-relevant statements offered a very close approximation of the extended reasoning approach.[22]

Extracting the terminological data, we found authoritative declarations of 434 functional properties, 57 inverse-functional properties, and 109 cardinality restrictions with a value of `1`.

We (again) gathered 11.93 million `owl:sameAs` statements, as well as 52.93 million memberships of inverse-functional properties, 11.09 million memberships of functional properties, and 2.56 million cardinality-relevant triples. Of these, respectively 22.14 million (41.8%), 1.17 million (10.6%) and 533 thousand (20.8%) were asserted—however, in the resulting closed `owl:sameAs` data derived with and without the extra reasoned triples, we detected a variation of less than 12 thousand terms (0.08%), where only 129 were URIs, and where other variations in statistics were less than 0.1% (e.g., there were 67 less equivalence classes when the reasoned triples were included).

From previous experience [22], we were aware of certain values for inverse-functional properties and functional properties which are erroneously published by exporters and which cause massive incorrect consolidation. We again blacklist statements featuring such values from our consolidation processing, where we give the top 10 such values encountered for our corpus in Table 5—this blacklist is the result of trial and error, manually inspecting large equivalence classes and the most common values for (inverse-)functional proper-

[22]At least in terms of pure *quantity*. However, we do not give an indication of the *quality* or *importance* of those few equivalences we miss with this approximation, which may be application specific.

#	**Blacklisted Term**	**Occurrences**
1	empty literals	584,735
2	`<http://null>`	414,088
3	`<http://www.vox.com/gone/>`	150,402
4	`"08445a31a78661b5c746feff39a9db6e4e2cc5cf"`	58,338
5	`<http://www.facebook.com>`	6,988
6	`<http://facebook.com>`	5,462
7	`<http://www.google.com>`	2,234
8	`<http://www.facebook.com/>`	1,254
9	`<http://google.com>`	1,108
10	`<http://null.com>`	542

Table 5
Top ten most frequently occurring blacklisted values

ties. Empty literals are commonly exported (with and without language tags) as values for inverse-functional-properties (particularly FOAF "chat-ID properties"). The literal `"08445a31a78661b5c746feff39a9db6e4e2cc5cf"` is the SHA-1 hash of the string '`mailto:`', commonly assigned as a `foaf:mbox_sha1sum` value to users who don't specify their email in some input form. The remaining URIs are mainly user-specified values for `foaf:homepage`, or values automatically assigned for users that don't specify such.[23]

During the computation of the `owl:sameAs` closure, we found zero inferences through cardinality rules, 106.8 thousand raw `owl:sameAs` inferences through function-property reasoning, and 8.7 million raw `owl:sameAs` inferences through inverse-functional-property reasoning. The final canonicalised, closed, and non-symmetric `owl:sameAs` index (such that s_1 SA s_2, $s_1 > s_2$, and s_2 is a canon) contained 12.03 million statements.

From this data, we generated 2.82 million equivalence classes (an increase of 1.31× from baseline consolidation) mentioning a total of 14.86 million terms (an increase of 2.58× from baseline—5.77% of all URIs and blank-nodes), of which 9.03 million were blank-nodes (an increase of 2173× from baseline—5.46% of all blank-nodes) and 5.83 million were URIs (an increase of 1.014× from baseline—6.33% of all URIs). Thus, we see a large expansion in the amount of blank-nodes consolidated, but only minimal expansion in the set of URIs referenced in the equivalence classes. With respect to the canonical identifiers, 641 thousand (22.7%) were blank-nodes and 2.18 million (77.3%) were URIs.

Figure 5 contrasts the equivalence class sizes for the baseline approach (seen previously in Figure 3), and for the extended reasoning approach. Overall, there is an observable increase in equivalence class sizes, where we see the average equivalence class size grow to 5.26 entities (1.98× baseline), the largest equivalence class size grow to 33,052 (3.9× baseline) and the percentage of equivalence classes with the minimum size 2 drop to 63.1% (from 74.1% in baseline).

In Table 6, we update the five largest equivalence classes. Result **2** carries over from the baseline consolidation. The rest of the results are largely intra-PLD equivalences, where the entity is described using thousands of blank-nodes, with a consistent (inverse-)functional property value attached. Result **1** refers to a meta-user—labelled `Team Vox`—commonly appearing in user-FOAF exports on the Vox blogging platform.[24] Result **3** refers to a person identified using blank-nodes (and once by URI) in thousands of RDF documents resident on the same server. Result **4** refers to the Image Bioinformatics Research Group in the University of Oxford—labelled `IBRG`—where again it is identified in thousands of documents using different blank-nodes, but a consistent `foaf:homepage`. Result **5** is similar to result **1**, but for a Japanese version of the Vox user.

Figure 6 presents a similar analysis to Figure 5, this time looking at identifiers on a PLD-level granularity. Interestingly, the difference between the two approaches is not so pronounced, initially indicating that many of the additional equivalences found through the consolidation rules are "intra-PLD". In the baseline consolidation approach, we determined that 57% of equivalence classes were inter-PLD (contain identifiers from more that one PLD), with the plurality of equivalence classes containing identifiers from precisely two PLDs (951 thousand, 44.1%); this indicates that explicit `owl:sameAs` relations are commonly asserted between PLDs. In the extended consolidation approach (which of course subsumes the above results), we determined that the percentage of inter-PLD equivalence classes dropped *to* 43.6%, with the majority of equivalence classes containing identifiers from only one PLD (1.59 million, 56.4%). The entity with the most diverse identifiers (the observable outlier on the x-axis in Figure 6) was the person "Dan Brickley"—one of the founders and leading contributors of the FOAF project—with 138 identifiers (67 URIs and 71 blank-nodes) minted in 47 PLDs; various other prominent community members and some country identifiers also featured high on the list.

[23]Our full blacklist contains forty-one such values, and can be found at `http://aidanhogan.com/swse/blacklist.txt`.

[24]This site shut down on 2010/09/30.

#	Canonical Term (Lexically Lowest in Equivalence Class)	Size	Correct?
1	`bnode37@http://a12iggymom.vox.com/profile/foaf.rdf`	33,052	✓
2	`http://bio2rdf.org/dailymed_drugs:1000`	8,481	×
3	`http://ajft.org/rdf/foaf.rdf#_me`	8,140	✓
4	`bnode4@http://174.129.12.140:8080/tcm/data/association/100`	4,395	✓
5	`bnode1@http://aaa977.vox.com/profile/foaf.rdf`	1,977	✓

Table 6
Largest 5 equivalence classes after extended consolidation

#	Canonical Identifier	BL#	R#
1	`<http://dblp.l3s.de/.../Tim_Berners-Lee>`	26	50
2	`<genid:danbri>`	10	138
3	`<http://update.status.net/>`	0	0
4	`<http://www.ldodds.com/foaf/foaf-a-matic>`	0	0
5	`<http://update.status.net/user/1#acct>`	0	6

Table 7
Equivalence class sizes for top five SWSE-ranked identifiers with respect to baseline (BL#) and reasoning (R#) consolidation

In Table 7, we compare the consolidation of the top five ranked identifiers in the SWSE system (see [24]). The results refer respectively to (i) the (co-)founder of the Web "Tim Berners-Lee"; (ii) "Dan Brickley" as aforementioned; (iii) a meta-user for the micro-blogging platform StatusNet which exports RDF; (iv) the "FOAF-a-matic" FOAF profile generator (linked from many diverse domains hosting FOAF profiles it created); and (v) "Evan Prodromou", founder of the `identi.ca`/StatusNet micro-blogging service and platform. We see a significant increase in equivalent identifiers found for these results; however, we also noted that after reasoning consolidation, Dan Brickley was conflated with a second person.[25]

Note that the most frequently co-occurring PLDs in our equivalence classes remained unchanged from Table 3.

During the rewrite of the main corpus, terms in 151.77 million subject positions (13.58% of all subjects) and 32.16 million object positions (3.53% of non-`rdf:type` objects) were rewritten, giving a total of 183.93 million positions rewritten (1.8× the baseline consolidation approach). In Figure 7, we compare the re-use of terms across PLDs before consolidation, after baseline consolidation, and after the extended reasoning consolidation. Again, although there is an increase in re-use of identifiers across PLDs, we note that: (i) the vast majority of identifiers (about 99%) still only appear in one PLD; (ii) the difference between the baseline and extended reasoning approach is not so pronounced. The most widely referenced consolidated entity—in terms of unique PLDs—was "Evan Prodromou" as aforementioned, referenced with six equivalent URIs in 101 distinct PLDs.

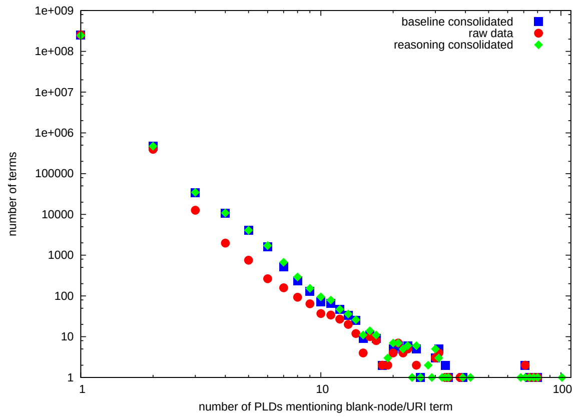


Fig. 7. Distribution of number of PLDs terms are referenced by, for the raw, baseline consolidated, and reasoning consolidated data (log/log)

In summary, we posit that applying the consolidation rules with respect to asserted data is a good approximation for our Linked Data corpus, and that in comparison to the baseline consolidation over explicit `owl:sameAs`, (i) the additional consolidation rules offer a large bulk of intra-PLD consolidation of blank-nodes with large equivalence-class sizes, which we believe to be due to publishing practices whereby a given exporter uses consistent inverse-functional property values instead of URIs to uniquely identify entities across local documents; and (ii) where there is only a minor expansion (1.014×) in the number of URIs involved in the consolidation.

6. Statistical Concurrence Analysis

In this section, we introduce methods for deriving a weighted concurrence score between entities in the Linked Data corpus: we define *entity concurrence* as the sharing of outlinks, inlinks and attribute values, denoting a specific form of similarity. We use these concurrence measures to materialise new links between such

[25] Domenico Gendarmi with three URIs—one document assigns one of Dan's `foaf:mbox_sha1sum` values (for `danbri@w3.org`) to Domenico: `http://foafbuilder.qdos.com/people/myriamleggieri.wordpress.com/foaf.rdf`

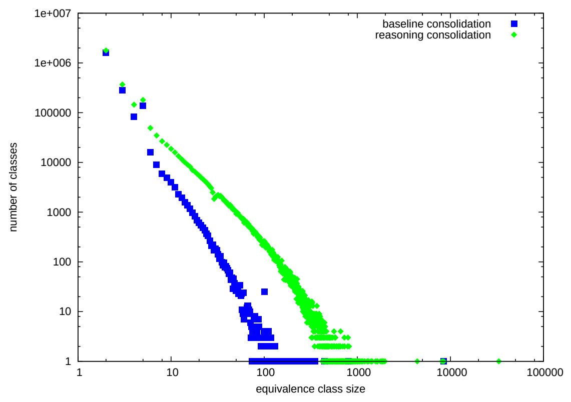


Fig. 5. Distribution of the number of identifiers per equivalence classes for baseline consolidation and extended reasoning consolidation (log/log)

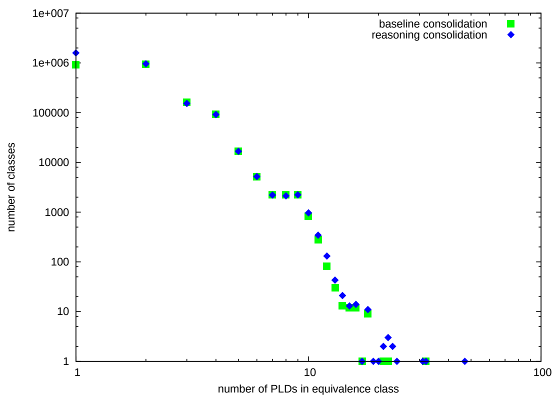


Fig. 6. Distribution of the number of PLDs per equivalence class for baseline consolidation and extended reasoning consolidation (log/log)

entities, and will also leverage the concurrence measures in Section 7 for disambiguating entities. The methods described herein are based on preliminary works we presented in [26], where we:

– investigated domain-agnostic statistical methods for performing consolidation and identifying equivalent entities;
– formulated an initial small-scale (5.6 million triples) evaluation corpus for the statistical consolidation using reasoning consolidation as a best-effort "gold-standard".

Our evaluation gave mixed results where we found some correlation between the reasoning consolidation and the statistical methods, but we also found that our methods gave incorrect results at high degrees of confidence for entities that were clearly not equivalent, but intuitively shared many links and attribute values in common. This of course highlights a crucial fallacy in our speculative approach: in almost all cases, even the highest degree of similarity/concurrence does not necessarily indicate equivalence or co-reference (cf. [20, Section 4.4]). Similar philosophical issues arise with respect to handling transitivity for the weighted "equivalences" derived [9,29].

However, deriving weighted concurrence measures has applications other than approximative consolidation: in particular, we can materialise named relationships between entities which share a lot in common, thus increasing the level of inter-linkage between entities in the corpus. Also, as we will see later, we can leverage the concurrence metrics to "rebuild" erroneous equivalence classes found during the disambiguation step. Thus, we present a modified version of the statistical analysis presented in [26], describe a (novel) scalable and distributed implementation thereof, and finally evaluate the approach with respect to finding highly-concurring entities in our 1 billion triple Linked Data corpus.

6.1. *High-level approach*

Our statistical concurrence analysis inherits similar *primary requirements* to that imposed for consolidation: the approach should be **scalable**, **fully automatic**, and **domain agnostic** to be applicable in our scenario. Similarly, with respect to *secondary criteria*, the approach should be **efficient to compute**, should give **high precision**, and should give **high recall**. Compared to consolidation, high precision is not as critical for our statistical use-case: in SWSE, we aim to use concurrency measures as a means of *suggesting* additional navigation steps for users browsing the entities—if the suggestion is uninteresting, it can be ignored, whereas incorrect consolidation will often lead to conspicuously garbled results, aggregating data on multiple disparate entities.

Thus, our requirements (particularly for scale) preclude the possibility of complex analyses or any form of pair-wise comparison, etc. Instead, we aim to design lightweight methods implementable by means of distributed sorts and scans over the corpus. Our methods are designed around the following intuitions and assumptions:

(i) the concurrency of entities is measured as a function of their *shared pairs*, be they predicate-subject (loosely, inlinks), or predicate-object pairs (loosely, outlinks or attribute values);
(ii) the concurrence measure should give a *higher weight to exclusive shared-pairs*—pairs which are typically shared by few entities, for edges (predicates) which typically have a low in-degree/out-

degree;

(iii) with the possible exception of correlated pairs, *each additional shared pair should increase the concurrency* of the entities—a shared pair cannot reduce the measured concurrency of the sharing entities;

(iv) *strongly exclusive property-pairs should be more influential than a large set of weakly exclusive pairs*;

(v) *correlation* may exist between shared pairs—e.g., two entities may share an inlink and an inverse-outlink to the same node (e.g., `foaf:depiction`, `foaf:depicts`), or may share a large number of shared pairs for a given property (e.g., two entities co-authoring one paper are more likely to co-author subsequent papers)—where we wish to dampen the cumulative effect of correlation in the concurrency analysis;

(vi) the *relative value* of the concurrency measure is important; the absolute value is unimportant.

In fact, the concurrency analysis follows a similar principle to that for consolidation, where instead of considering discrete functional and inverse-functional properties as given by the semantics of the data, we attempt to identify properties which are quasi-functional, quasi-inverse-functional, or what we more generally term *exclusive*: we determine the degree to which the values of properties (here abstracting directionality) are unique to an entity or set of entities. The concurrency between two entities then becomes an aggregation of the weights for the property-value pairs they share in common.

To take a running example, consider the following data:

```
dblp:AliceB10 foaf:maker ex:Alice .
dblp:AliceB10 foaf:maker ex:Bob .

ex:Alice foaf:gender "female" .
ex:Alice foaf:workplaceHomepage <http://wonderland.com> .

ex:Bob foaf:gender "male" .
ex:Bob foaf:workplaceHomepage <http://wonderland.com> .

ex:Claire foaf:gender "female" .
ex:Claire foaf:workplaceHomepage <http://wonderland.com> .
```

where we want to determine the level of (relative) concurrency between three colleagues: `ex:Alice`, `ex:Bob` and `ex:Claire`: i.e., how much do they coincide/concur with respect to exclusive shared pairs.

6.1.1. *Quantifying concurrence*

First, we want to characterise the uniqueness of properties; thus, we analyse their *observed* cardinality and inverse-cardinality as found in the corpus (in contrast to their defined cardinality as possibly given by the formal semantics):

Definition 1 (Observed Cardinality) *Let G be an RDF graph, p be a property used as a predicate in G and s be a subject in G. The* observed cardinality *(or henceforth in this section, simply* cardinality*) of p* wrt *s* in *G, denoted $\mathrm{Card}_G(p, s)$, is the cardinality of the set $\{o \in \mathsf{C} \mid (s, p, o) \in G\}$.*

Definition 2 (Observed Inverse-Cardinality) *Let G and p be as before, and let o be an object in G. The* observed inverse-cardinality *(or henceforth in this section, simply* inverse-cardinality*) of p* wrt *o* in *G, denoted $\mathrm{ICard}_G(p, o)$, is the cardinality of the set $\{s \in \mathsf{U} \cup \mathsf{B} \mid (s, p, o) \in G\}$.*

Thus, loosely, the observed cardinality of a property-subject pair is the number of unique objects it appears with in the graph (or unique triples it appears in); letting G_{ex} denote our example graph, then, e.g., $\mathrm{Card}_{G_{ex}}(\texttt{foaf:maker}, \texttt{dblp:AliceB10}) = 2$. We see this value as a good indicator of how *exclusive* (or *selective*) a given property-subject pair is, where sets of entities appearing in the object position of low-cardinality pairs are considered to concur more than those appearing with high-cardinality pairs. The observed inverse-cardinality of a property-object pair is the number of unique subjects it appears with in the graph—e.g., $\mathrm{ICard}_{G_{ex}}(\texttt{foaf:gender}, \texttt{"female"}) = 2$. Both directions are considered analogous for deriving concurrence scores—note however that we do not consider concurrence for literals (i.e., we do not derive concurrence for literals which share a given predicate-subject pair; we do of course consider concurrence for subjects with the same literal value for a given predicate).

To avoid unnecessary duplication, we henceforth focus on describing only the inverse-cardinality statistics of a property, where the analogous metrics for plain-cardinality can be derived by switching subject and object (that is, switching directionality)—we choose the inverse direction as perhaps being more intuitive, indicating concurrence of entities in the subject position based on predicate-object pairs they share.

Definition 3 (Average Inverse-Cardinality) *Let G be an RDF graph, and p be a property used as a predicate in G. The* average inverse-cardinality *of p, written $\mathrm{AIC}_G(p)$, is the average of the non-zero inverse-cardinalities of p in the graph G. Formally:*

$$\mathrm{AIC}_G(p) = \frac{|\{(s, o) \mid (s, p, o) \in G\}|}{|\{o \mid \exists s : (s, p, o) \in G\}|}$$

.

The average cardinality of a property is defined analogously. Note that the (inverse-)cardinality value of any term appearing as a predicate in the graph is necessarily greater-than or equal-to one: the numerator is by definition greater-than or equal-to the denominator. Taking an example, $\mathrm{AIC}_{G_{ex}}(\texttt{foaf:gender}) = 1.5$, which can be viewed equivalently as the average non-zero cardinalities of `foaf:gender` (1 for `"male"` and 2 for `"female"`), or the number of triples with predicate `foaf:gender` divided by the number of unique values appearing in the object position of such triples ($\frac{3}{2}$).

We call a property p for which we observe $\mathrm{AIC}_G(p) \approx 1$, a *quasi-inverse-functional property* with respect to the graph G, and analogously properties for which we observe $\mathrm{AC}_G(p) \approx 1$ as *quasi-functional properties*. We see the values of such properties—in their respective directions—as being very exceptional: very rarely shared by entities. Thus, we would expect a property such as `foaf:gender` to have a high $\mathrm{AIC}_G(p)$ since there are only two object-values (`"male"`, `"female"`) shared by a large number of entities, whereas we would expect a property such as `foaf:workplaceHomepage` to have a lower $\mathrm{AIC}_G(p)$ since there are arbitrarily many values to be shared amongst the entities; given such an observation, we then surmise that a shared `foaf:gender` value represents a weaker "indicator" of concurrence than a shared value for `foaf:workplaceHomepage`.

Given that we deal with incomplete information under the Open World Assumption underlying RDF(S)/OWL, we also wish to weight the average (inverse) cardinality values for properties with a low number of observations towards a global mean—consider a fictional property `ex:maritalStatus` for which we only encounter a few predicate-usages in a given graph, and consider two entities given the value `"married"`: given sparse inverse-cardinality observations, we may naïvely over-estimate the significance of this property-object pair as an indicator for concurrence. Thus, we use a credibility formula as follows to weight properties with few observations towards a global mean:

Definition 4 (Adjusted Average Inverse-Cardinality) *Let p be a property appearing as a predicate in the graph G. The* adjusted average cardinality of p with respect to G *is then*

$$\mathrm{AAIC}_G(p) = \frac{\mathrm{AIC}_G(p) \times |G^{\overrightarrow{p}}| + \overline{\mathrm{AIC}_G} \times |G^{\rightarrow}|}{|G^{\overrightarrow{p}}| + |G^{\rightarrow}|} \quad (1)$$

where $|G^{\overrightarrow{p}}|$ is the number of distinct objects that appear in a triple with p as a predicate (the denominator of Definition 3), $\overline{\mathrm{AIC}_G}$ is the average inverse-cardinality for all predicate-object pairs (formally, $\overline{\mathrm{AIC}_G} = \frac{|G|}{|\{(p,o) \mid \exists s : (s,p,o) \in G\}|}$), and $|G^{\rightarrow}|$ is the average number of distinct objects for all predicates in the graph (formally, $|G^{\rightarrow}| = \frac{|\{(p,o) \mid \exists s : (s,p,o) \in G\}|}{|\{p \mid \exists s, \exists o : (s,p,o) \in G\}|}$)

Some reduction is possible, following $\mathrm{AIC}_G(p) \times |G^{\overrightarrow{p}}| = |\{(s,o) \mid (s,p,o) \in G\}|$ denoting the number of triples for which p appears as a predicate in graph G, and $\overline{\mathrm{AIC}_G} \times |G^{\rightarrow}| = \frac{|G|}{|\{p \mid \exists s, \exists o : (s,p,o) \in G\}|}$, denoting the average number of triples per predicate. We maintain Equation 1 in the given unreduced form as it more clearly corresponds to the structure of a standard credibility formula: the reading ($\mathrm{AIC}_G(p)$) is dampened towards a mean ($\overline{\mathrm{AIC}_G}$) by a factor determined by the size of the sample used to derive the reading ($|G^{\overrightarrow{p}}|$) relative to the average sample size ($|G^{\rightarrow}|$).

Now, we move towards combining these metrics to determine the concurrency of entities who share a given non-empty set of property-value pairs. To do so, we combine the adjusted average (inverse) cardinality values which apply generically to properties, and the (inverse) cardinality values which apply to a given property-value pair. For example, take the property `foaf:workplaceHomepage`: entities that share a value referential to a large company—e.g., `http://google.com/`—should not gain as much concurrence as entities that share a value referential to a smaller company—e.g., `http://deri.ie/`. Conversely, consider a fictional property `ex:citizenOf`—which relates a citizen to its country—for which we find many observations in our corpus, returning a high AAIC value, and consider that only two entities share the value `ex:Vanuatu` for this property: given that our data are incomplete, we can use the high AAIC value of `ex:citizenOf` to determine that the property is usually not exclusive, and that it is generally not a good indicator of concurrence.[26]

We start by assigning a coefficient to each pair (p,o) and each pair (p,s) that occur in the dataset, where the coefficient is an indicator of how exclusive that pair is:

Definition 5 (Concurrence Coefficients) *The* concurrence-coefficient of a predicate-subject pair (p,s) *with respect to a graph G is given as:*

$$\mathrm{C}_G(p,s) = \frac{1}{\mathrm{Card}_G(p,s) \times \mathrm{AAC}_G(p)}$$

[26]Here, we try to distinguish between property-value pairs which are exclusive in reality (i.e., on the level of what's signified) and those which are exclusive in the given graph. Admittedly, one could think of counter-examples where not including the general statistics of the property may yield a better indication of weighted concurrence, particularly for generic properties which can be applied in many contexts; for example, consider the exclusive predicate-object pair (`skos:subject`, `category:Koenigsegg_vehicles`) given for a non-exclusive property.

and the concurrence-coefficient of a predicate-object pair (p, o) *with respect to a graph* G *is analogously given as:*

$$\mathrm{IC}_G(p, o) = \frac{1}{\mathrm{ICard}_G(p, o) \times \mathrm{AAIC}_G(p)}$$

Again, please note that these coefficients fall into the interval $]0, 1]$ since the denominator, by definition, is necessarily greater than one.

To take an example, let p_{wh} = `foaf:workplaceHomepage` and say that we compute $\mathrm{AAIC}_G(p_{wh})$ = 7 from a large number of observations, indicating that each workplace homepage in the graph G is linked to by, on average, seven employees. Further, let o_g = `http://google.com/` and assume that o_g occurs 2,000 times as a value for p_{wh}: $\mathrm{ICard}_G(p_{wh}, o_g)$ = 2,000; now, $\mathrm{IC}_G(p_{wh}, o_g) = \frac{1}{2{,}000 \times 7} = 0.00007$. Also, let o_d = `http://deri.ie/` such that $\mathrm{ICard}_G(p_{wh}, o_d)$ = 100; now, $\mathrm{IC}_G(p_{wh}, o_d) = \frac{1}{10 \times 7} \approx 0.00143$. Here, sharing DERI as a workplace will indicate a higher level of concurrence than analogously sharing Google.

Finally, we require some means of aggregating the coefficients of the set of pairs that two entities share to derive the final concurrence measure.

Definition 6 (Aggregated Concurrence Score) *Let* $Z = (z_1, \ldots z_n)$ *be a tuple such that for each* $i = 1, \ldots, n$, $z_i \in]0, 1]$. *The aggregated concurrence value* ACS_n *is computed iteratively: starting with* $\mathrm{ACS}_0 = 0$, *then for each* $k = 1 \ldots n$, $\mathrm{ACS}_k = z_k + \mathrm{ACS}_{k-1} - z_k * \mathrm{ACS}_{k-1}$.

The computation of the ACS value is the same process as determining the probability of two independent events occurring—$P(A \vee B) = P(A) + P(B) - P(A * B)$—which is by definition commutative and associative, and thus computation is independent of the order of the elements in the tuple. It may be more accurate to view the coefficients as *fuzzy values*, and the aggregation function as a disjunctive combination in some extensions of fuzzy logic [54].

However, the underlying coefficients may not be derived from strictly independent phenomena: there may indeed be correlation between the property-value pairs that two entities share. To illustrate, we reintroduce a relevant example from [26] shown in Figure 8, where we see two researchers that have co-authored many papers together, have the same affiliation, and are based in the same country.

This example illustrates three categories of concurrence correlation:

(i) *same-value correlation* where two entities may be linked to the same value by multiple predicates in either direction (e.g., `foaf:made`, `dc:creator`, `swrc:author`, `foaf:maker`);

(ii) *intra-property correlation* where two entities which share a given property-value pair are likely to share further values for the same property (e.g., co-authors sharing one value for `foaf:made` are more likely to share further values);

(iii) *inter-property correlation* where two entities sharing a given property-value pair are likely to share further distinct but related property-value pairs (e.g., having the same value for `swrc:affiliation` and `foaf:based_near`).

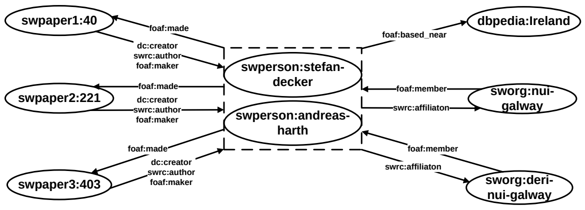


Fig. 8. Example of same-value, inter-property and intra-property correlation, where the two entities under comparison are highlighted in the dashed box, and where the labels of inward-edges (with respect to the principal entities) are italicised and underlined (from [26])

Ideally, we would like to reflect such correlation in the computation of the concurrence between the two entities.

Regarding *same-value correlation*, for a value with multiple edges shared between two entities, we choose the shared predicate edge with the lowest AA[I]C value and disregard the other edges: i.e., we only consider the most exclusive property used by both entities to link to the given value and prune the other edges.

Regarding *intra-property correlation*, we apply a lower-level aggregation for each predicate in the set of shared predicate-value pairs. Instead of aggregating a single tuple of coefficients, we generate a bag of tuples $\mathbf{Z} = \{Z_{p_1}, \ldots, Z_{p_n}\}$, where each element Z_{p_i} represents the tuple of (non-pruned) coefficients generated for the predicate p_i.[27] We then aggregate this bag as follows:

$$\mathrm{ACS}(\mathbf{Z}) = \mathrm{ACS}(\mathrm{ACS}(Z_{p_i} * \mathrm{AA[I]C}(p_i))_{Z_{p_i} \in \mathbf{Z}})$$

where AA[I]C is either AAC or AAIC, dependant on the directionality of the predicate-value pair observed. Thus, the total contribution possible through a given predicate (e.g., `foaf:made`) has an upper-bound set as its AA[I]C value, where each successive shared value for that predicate (e.g., each successive co-authored paper) contributes positively (but increasingly less) to the overall concurrence measure.

[27]For brevity, we omit the graph subscript.

Detecting and counteracting *inter-property correlation* is perhaps more difficult, and we leave this as an open question.

6.1.2. *Implementing entity-concurrence analysis*

We aim to implement the above methods using sorts and scans, and wish to avoid any form of complex indexing, or pair-wise comparison. Firstly, we wish to extract the statistics relating to the (inverse-)cardinalities of the predicates in the data. Given that there are 23 thousand unique predicates found in the input corpus, we assume that we can fit the list of predicates and their associated statistics in memory—if such were not the case, one could consider an on-disk map, where we would expect a high cache hit-rate based on the distribution of property occurrences in the data (cf. [24]).

Moving forward, we can calculate the necessary predicate-level statistics by first sorting the data according to natural order (s, p, o, c), and then scanning the data, computing the cardinality (number of distinct objects) for each (s, p) pair, and maintaining the average cardinality for each p found. For inverse-cardinality scores, we apply the same process, sorting instead by (o, p, s, c) order, counting the number of distinct subjects for each (p, o) pair, and maintaining the average inverse-cardinality scores for each p. After each scan, the statistics of the properties are adjusted according to the credibility formula in Equation 4.

We then apply a second scan of the sorted corpus; first we scan the data sorted in natural order, and for each (s, p) pair, for each set of unique objects O_{ps} found thereon, and for each pair in

$$\{(o_i, o_j) \in \mathsf{U} \cup \mathsf{B} \times \mathsf{U} \cup \mathsf{B} \mid o_i, o_j \in O_{ps}, o_i < o_j\}$$

where $<$ denotes lexicographical order, we output the following sextuple to an on-disk file:

$$(o_i, o_j, \mathrm{C}(p, s), p, s, -)$$

where $\mathrm{C}(p, s) = \frac{1}{|O_{ps}| \times \mathrm{AAC}(p)}$. We apply the same process for the other direction, outputting analogous sextuples of the form:

$$(s_i, s_j, \mathrm{IC}(p, o), p, o, +)$$

We call the sets O_{ps} are their analogues S_{po} *concurrence classes*, denoting sets of entities which share the given predicate-subject/predicate-object pair respectively. Here, note that the '$+$' and '$-$' elements simply demarcate and track the directionality from which the tuple was generated, required for the final aggregation of the co-efficient scores. Similarly, we do not immediately materialise the symmetric concurrence scores, where we instead do so at the end so as to forego duplication of intermediary processing.

Once generated, we can sort the two files of tuples by their natural order, and perform a merge-join on the first two elements—generalising the directional o_i/s_i to simply e_i, each (e_i, e_j) pair denotes two entities which share some predicate-value pairs in common, where we can scan the sorted data and aggregate the final concurrence measure for each (e_i, e_j) pair using the information encoded in the respective tuples. We can thus generate (trivially sorted) tuples of the form (e_i, e_j, s), where s denotes the final aggregated concurrence score computed for the two entities; optionally, we can also write the symmetric concurrence tuples (e_j, e_i, s) which can be sorted separately as required.

Note that the number of tuples generated is quadratic with respect to the size of the respective concurrence class, which becomes a major impediment for scalability given the presence of large such sets—for example, consider a corpus containing 1 million persons sharing the value `"female"` for the property `foaf:gender`, where we would have to generate $\frac{10^{6 \times 2} - 10^6}{2} \approx 500$ billion non-reflexive, non-symmetric concurrence tuples. However, we can leverage the fact that such sets can only invoke a minor influence on the final concurrence of their elements, given that the magnitude of the set—e.g., $|S_{po}|$—is a factor in the denominator of the computed $\mathrm{C}(p, o)$ score, such that $\mathrm{C}(p, o) \propto \frac{1}{|S_{op}|}$. Thus, in practice, we implement a maximum-size threshold for the S_{po} and O_{ps} concurrence classes: this threshold is selected based on a practical upper limit for raw similarity tuples to be generated, where the appropriate maximum class size can trivially be determined alongside the derivation of the predicate statistics. For the purpose of evaluation, we choose to keep the number of raw tuples generated at around $\sim$1 billion, and so set the maximum concurrence class size at 38—we will see more in Section 6.4.

6.2. *Distributed implementation*

Given the previous discussion, our distributed implementation is fairly straight-forward as follows:

(i) **coordinate**: the slave machines split their segment of the corpus according to a modulo-hash function on the subject position of the data, sort the segments, and send the split segments to the peer determined by the hash-function; the slaves simultaneously gather incoming sorted segments, and subsequently perform a merge-sort of the segments;

(ii) **coordinate**: the slave machines apply the same operation, this time hashing on object—triples with `rdf:type` as predicate are not included in the object-hashing; subsequently the slaves merge-sort the segments ordered by object;

(iii) **run**: the slave machines then extract predicate-level statistics, and statistics relating to the concurrence-class-size distribution which are used to decide upon the class size threshold;

(iv) **gather**/**flood**/**run**: the master machine gathers and aggregates the high-level statistics generated by the slave machines in the previous step and sends a copy of the global statistics back to each machine; the slaves subsequently generate the raw concurrence-encoding sextuples as described before from a scan of the data in both orders;

(v) **coordinate**: the slave machines coordinate the locally generated sextuples according to the first element (join position) as before;

(vi) **run**: the slave machines aggregate the sextuples coordinated in the previous step, and produce the final non-symmetric concurrence tuples;

(vii) **run**: the slave machines produce the symmetric version of the concurrence tuples, and coordinate and sort on the first element.

Here, we make heavy use of the **coordinate** function to align data according to the join position required for the subsequent processing step—in particular, aligning the raw data by subject and object, and then the concurrence tuples analogously.

Note that we do not hash on the object position of `rdf:type` triples: our raw corpus contains 206.8 million such triples, and given the distribution of class memberships, we assume that hashing these values will lead to uneven distribution of data, and subsequently uneven load balancing—e.g., 79.2% of all class memberships are for `foaf:Person`, hashing on which would send 163.7 million triples to one machine, which alone is greater than the average number of triples we would expect per machine (139.8 million). In any case, given that our corpus contains 105 thousand unique values for `rdf:type`, we would expect the average-inverse-cardinality to be approximately 1,970—even for classes with two members, the potential effect on concurrence is negligible.

6.3. *Performance evaluation*

We apply our concurrence analysis over the consolidated corpus derived in Section 5. The total time taken was 13.9 h. Sorting, splitting and scattering the data according to subject on the slave machines took 3.06 h, with an average idle time of 7.7 min (4.2%). Subsequently, merge-sorting the sorted segments took 1.13 h, with an average idle time of 5.4 min (8%). Analogously sorting, splitting and scattering the non-`rdf:type` statements by object took 2.93 h, with an average idle time of 11.8 min (6.7%). Merge sorting the data by object took 0.99 h, with an average idle time of 3.8 min (6.3%). Extracting the predicate statistics and threshold information from data sorted in both orders took 29 min, with an average idle time of 0.6 min (2.1%). Generating the raw, unsorted similarity tuples took 69.8 min with an average idle time of 2.1 min (3%). Sorting and coordinating the raw similarity tuples across the machines took 180.1 min, with an average idle time of 14.1 min (7.8%). Aggregating the final similarity took 67.8 min, with an average idle time of 1.2 min (1.8%).

Table 8 presents a breakdown of the timing of the task. Although this task requires some aggregation of global-knowledge by the master machine, the volume of data involved is minimal: a total of 2.1 minutes is spent on the master machine performing various minor tasks (initialisation, remote calls, logging, aggregation and broadcast of statistics). Thus, 99.7% of the task is performed in parallel on the slave machine. Although there is less time spent waiting for the master machine compared to the previous two tasks, deriving the concurrence measures involves three expensive sort/**coordinate**/merge-sort operations to redistribute and sort the data over the slave swarm. The slave machines were idle for, on average, 5.8% of the total task time; most of this idle time (99.6%) was spent waiting for peers. The master machines was idle for almost the entire task, with 99.7% waiting for the slave machines to finish their tasks—again, interleaving a job for another task would have practical benefits.

6.4. *Results Evaluation*

With respect to data distribution, after hashing on subject we observed an average absolute deviation (average distance from the mean) of 176 thousand triples across the slave machines, representing an average 0.13% deviation from the mean: near-optimal data distribution. After hashing on the object of non-`rdf:type` triples, we observed an average absolute deviation of 1.29 million triples across the machines, representing an average 1.1% deviation from the mean; in particular, we note that one machine was assigned 3.7 million triples above the mean (an additional 3.3% above the mean). Although not optimal, the percentage of data deviation

Category	**min**	**% Total**
Total execution time	835.4	100
Master (Local)		
Executing	2.1	0.3
Miscellaneous	2.1	0.3
Idle (waiting for slaves)	833.3	99.7
Slave (Parallel)		
Avg. Executing (total exc. idle)	786.6	94.2
Split/sort/scatter (subject)	175.9	21.1
Merge-sort (subject)	62.4	7.5
Split/sort/scatter (object)	164	19.6
Merge-sort (object)	55.6	6.6
Extract High-level Statistics	28.4	3.3
Generate Raw Concurrence Tuples	67.7	8.1
Cooordinate/Sort Concurrence Tuples	166	19.9
Merge-sort/Aggregate Similarity	66.6	8
Avg. Idle	48.8	5.8
Waiting for peers	46.7	5.6
Waiting for master	2.1	0.3

Table 8
Breakdown of timing of distributed concurrence analysis

given by hashing on object is still within the natural variation in run-times we have seen for the slave machines during most parallel tasks.

In Figures 9(a) and 9(b), we illustrate the effect of including increasingly large concurrence classes on the number of raw concurrence tuples generated. For the predicate-object pairs, we observe a power-law relationship between the size of the concurrence class and the number of such classes observed. Second, we observe that the number of concurrences generated for each increasing class size initially remains fairly static—i.e., larger class sizes give quadratically more concurrences, but occur polynomially less often—until the point where the largest classes which generally only have one occurrence is reached, and the number of concurrences begins to increase quadratically. Also shown is the cumulative count of concurrence tuples generated for increasing class sizes, where we initially see a power-law correlation, which subsequently begins to flatten as the larger concurrence classes become more sparse (although more massive).

For the predicate-subject pairs, the same roughly holds true, although we see fewer of the very largest concurrence classes: the largest concurrence class given by a predicate-subject pair was 79 thousand, versus 1.9 million for the largest predicate-object pair, respectively given by the pairs (`kwa:map`, `macs:manual_rameau_lcsh`) and (`opiumfield:rating`, `""`). Also, we observe some "noise" where for milestone concurrence class sizes (esp., at 50, 100, 1,000, 2,000) we observe an unusual amount of classes. For example, there were 72 thousand

Predicate	**Objects**	**Triples**	**AAC**
`foaf:nick`	150,433,035	150,437,864	1.000
`lldpubmed:journal`	6,790,426	6,795,285	1.003
`rdf:value`	2,802,998	2,803,021	1.005
`eurostat:geo`	2,642,034	2,642,034	1.005
`eurostat:time`	2,641,532	2,641,532	1.005

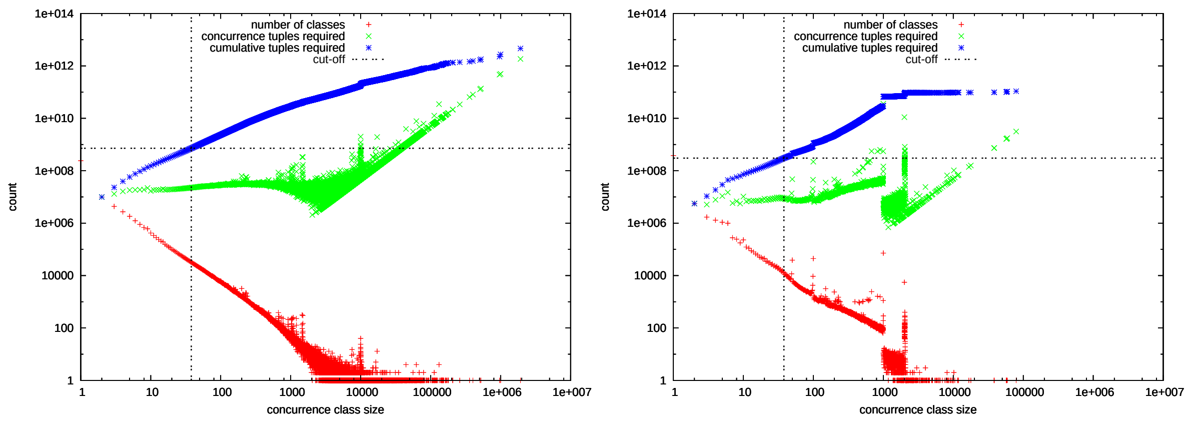
Table 9
Top five predicates with respect to lowest adjusted average cardinality (AAC)

concurrence classes of precisely size 1,000 (versus 88 concurrence classes at size 996)—the 1,000 limit was due to a FOAF exporter from the `hi5.com` which seemingly enforces that limit on the total "friends count" of users, translating into many users with precisely 1,000 values for `foaf:knows`.[28] Also for example, there were 5.5 thousand classes of size 2,000 (versus 6 classes of size 1,999)—almost all of these were due to an exporter from the `bio2rdf.org` domain which puts this limit on values for the `bio2rdf:linkedToFrom` property.[29] We also encountered unusually large numbers of classes approximating these milestones, such as 73 at 2,001. Such phenomena explain the staggered "spikes"and "discontinuities" in Figure 9(b), which can be observed to correlate with such milestone values (in fact, similar but less noticeable spikes are also present in Figure 9(a)).

These figures allow us to choose a threshold of concurrence-class size given an upper bound on raw concurrence tuples to generate. For the purposes of evaluation, we choose to keep the number of materialised concurrence tuples at around 1 billion, which limits our maximum concurrence class size to 38 (from which we produce 1.023 billion tuples: 721 million through shared (p, o) pairs and 303 million through (p, s) pairs.

With respect to the statistics of predicates, for the predicate-subject pairs, each predicate had an average of 25,229 unique objects for 37,953 total triples, giving an average cardinality of $\sim$1.5. We give the five predicates observed to have the lowest adjusted average cardinality in Table 9; note that two of these properties will not generate any concurrences since they are perfectly unique to a given object. For the predicate-object pairs, there was an average of 11,572 subjects for 20,532 triples, giving an average inverse-cardinality of $\sim$2.64; We give the five predicates observed to have the lowest adjusted average inverse cardinality in Table 10; again, four of these properties will not generate any concurrences since they are perfectly unique to a given subject.

[28] cf. `http://api.hi5.com/rest/profile/foaf/100614697`
[29] cf. `http://bio2rdf.org/mesh:D000123Q000235`



(a) Predicate-object pairs (b) Predicate-subject pairs

Fig. 9. Breakdown of potential concurrence classes given with respect to predicate-object pairs and predicate-subject pairs respectively, where for each class size on the x-axis (s_i) we show the number of classes (c_i); the raw non-reflexive, non-symmetric concurrence tuples generated ($sp_i = c_i \times \frac{s_i^2+s_i}{2}$); a cumulative count of tuples generated for increasing class sizes ($\widehat{sp_i} = \sum_{j \leq i} sp_j$); and our cut-off ($s_i = 38$) which we've chosen to keep the total number of tuples at $\sim$1 billion (log/log)

Predicate	**Subjects**	**Triples**	**AAIC**
`lldpubmed:meshHeading`	2,121,384	2,121,384	1.009
`opiumfield:recommendation`	1,920,992	1,920,992	1.010
`fb:type.object.key`	1,108,187	1,108,187	1.017
`foaf:page`	1,702,704	1,712,970	1.017
`skipinions:hasFeature`	1,010,604	1,010,604	1.019

Table 10
Top five predicates with respect to lowest adjusted average inverse-cardinality (AAIC)

Entity Label 1	**Entity Label 2**	**Concur**
1 `New York City`	`New York State`	791
2 `London`	`England`	894
3 `Tokyo`	`Japan`	900
4 `Toronto`	`Ontario`	418
5 `Philadelphia`	`Pennsylvania`	217

Table 11
Top five concurrent entities and the number of pairs they share

Aggregation produced a final total of 636.9 million weighted concurrence pairs, with a mean concurrence weight of ∼0.0159. Of these pairs, 19.5 million involved a pair of identifiers from different PLDs (3.1%), whereas 617.4 million involved identifiers from the same PLD; however, the average confidence value for an intra-PLD pair was 0.446, versus 0.002 for inter-PLD pairs—although fewer intra-PLD concurrences are found, they typically have higher confidences. [30]

In Table 11, we give the labels of top five most concurrent entities, including the number of pairs they share—the confidence score for each of these pairs was > 0.9999999. We note that they are all locations, where particularly on WIKIPEDIA (and thus filtering through to DBpedia), properties with location values are typically duplicated (e.g., `dbp:deathPlace`, `dbp:birthPlace`, `dbp:headquarters`—properties that are *quasi-functional*); for example, `New York City` and `New York State` are both the `dbp:deathPlace` of `dbpedia:Isacc_Asimov`, etc.

In Table 12, we give a description of the concurrent entities found for the top-five ranked entities—for brevity, again we show entity labels. In particular, we note that a large amount of concurrent entities are identified for the highly-ranked persons. With respect to the strongest concurrences: (i) Tim and his former student Lalana share twelve primarily academic links, coauthoring six papers; (ii) Dan and Libby, co-founders of the FOAF project, share 87 links, primarily 73 `foaf:knows` relations to and from the same people, as well as a co-authored paper, occupying the same professional positions, etc.; [31] (iii) `update.status.net` and `socialnetwork.ro` share a single `foaf:accountServiceHomepage` link from a common user; (iv) similarly, the FOAF-a-matic and `foaf.me` services share a single `mvcb:generatorAgent` inlink; (v) finally, Evan and Stav share 69 `foaf:knows` inlinks and outlinks exported from the `identi.ca` service.

[30] Note that we apply this analysis over the consolidated data, and thus this is an approximative reading for the purposes of illustration: we extract the PLDs from canonical identifiers, which are choosen based on arbitrary lexical ordering.

[31] Notably, Leigh Dodds (creator of the FOAF-a-matic service) is linked by the property `quaffing:drankBeerWith` to both.

Ranked Entity	#Con.	"Closest" Entity	Val.
`Tim Berners-Lee`	908	`Lalana Kagal`	0.83
`Dan Brickley`	2,552	`Libby Miller`	0.94
`update.status.net`	11	`socialnetwork.ro`	0.45
`FOAF-a-matic`	21	`foaf.me`	0.23
`Evan Prodromou`	3,367	`Stav Prodromou`	0.89

Table 12
Breakdown of concurrences for top five ranked entities, ordered by rank, with, respectively, entity label, number of concurrent entities found, the label of the concurrent entity with the largest degree, and finally the degree value

7. Entity Disambiguation

We have already seen that—even by only exploiting the formal logical consequences of the data through reasoning—consolidation may already be imprecise. Herein, we investigate an approach to identify 'incorrect' consolidation by means of inconsistency analysis, and a subsequent repair strategy based on statistical concurrence scores previously outlined.

7.1. *High-level approach*

The high-level approach is to see if the consolidation of any entities conducted in the previous step lead to any *novel* inconsistencies, and subsequently *recant* the equivalences involved; thus, it is important to note that our aim is not to repair inconsistencies in the data—we refer the interested reader to [4] for some previous works on this topic—but instead to repair incorrect consolidation symptomised by inconsistency. Herein, we aim to (i) describe what forms of inconsistency we detect and how we detect them; (ii) characterise how inconsistencies can be caused by consolidation using examples from our corpus where possible; (iii) discuss the repair of equivalence classes which have been determined to cause inconsistency.

First, in order to track which data are consolidated and which not, in the previous consolidation step we output sextuples of the form:

$$(s, p, o, c, s', o')$$

where s, p, o, c, denote the consolidated quadruple containing canonical identifiers in the subject/object position as appropriate, and s' and o' denote the input identifiers prior to consolidation.[32]

[32]We use syntactic shortcuts in our file to denote when $s = s'$ and/or $o = o'$. Maintaining the additional rewrite information during the consolidation process is trivial, where the output of consolidating subjects gives quintuples (s, p, o, c, s'), which are then sorted and consolidated by o to produce the given sextuples.

To detect inconsistencies in the consolidated corpus, we use the OWL 2 RL/RDF rules with the `false` consequent [17] as listed in Table B.4. Again, we italicise the labels of rules requiring new OWL 2 constructs, where we expect few such axioms to appear on the Web: a quick check of our corpus revealed that one document provides 8 `owl:AsymmetricProperty` and 10 `owl:IrreflexiveProperty` axioms[33], and one directory gives 9 `owl:AllDisjointClasses` axioms[34], and where we found no other OWL 2 axioms relevant to the rules in Table B.4. In any case, we include all rules with the exception of **cls-maxqc1**, where as we will see in Section 7.2, this rule is incompatible with our implementation which requires a consistent assertional join variable for computing a merge-join operation—in any case, again we found no such axoims in our corpus.

We also consider an additional rule, whose semantics are indirectly axiomatised by the OWL 2 RL/RDF rules (through **prp-fp**, **dt-diff** and **eq-diff1**), but which we must support directly since we do not consider consolidation of literals:

$\underline{\texttt{?p a owl:FunctionalProperty .}}$
`?x ?p ?l`$_1$ `, ?l`$_2$ `.`
`?l`$_1$ *`owl:differentFrom`* `?l`$_2$ `.`
$\Rightarrow$ `false`

where we underline the terminological pattern. To illustrate, we take an example from our corpus:

```
# Terminological [http://dbpedia.org/data3/length.rdf]
 dpo:length rdf:type owl:FunctionalProperty .
# Assertional [http://dbpedia.org/data/Fiat_Nuova_500.xml]
 dbpedia:Fiat_Nuova_500' dpo:length "3.546"^^xsd:double .
# Assertional [http://dbpedia.org/data/Fiat_500.xml]
 dbpedia:Fiat_Nuova' dpo:length "2.97"^^xsd:double .
```

(Note that herein, we will use the prime symbol [$'$] to denote identifiers considered coreferent by consolidation.) Here we see two very closely related models of cars consolidated in the previous step, but we now identify that they have two different values for `dpo:length`—a functional-property—and thus the consolidation raises an inconsistency.

Note that we do not expect the *`owl:differentFrom`* assertion to be materialised, but instead intend a rather more relaxed semantics based on a heurisitic comparison: given two (distinct) literal bindings for `?l`$_1$ and `?l`$_2$, we flag an inconsistency iff (i) the data values of the two bindings are not equal (standard OWL semantics); *and* (ii) their lower-case string value (minus language-tags and datatypes) are lexically unequal.

[33]`http://models.okkam.org/ENS-core-vocabulary#country_of_residence`
[34]`http://ontologydesignpatterns.org/cp/owl/fsdas/`

In particular, the relaxation is inspired by the definition of the FOAF (datatype) functional properties `foaf:age`, `foaf:gender`, and `foaf:birthday`, where the range of these properties is rather loosely defined: a generic range of `rdfs:Literal` is formally defined for these properties, with informal recommendations to use `male`/`female` as gender values, and `MM-DD` syntax for birthdays, but not giving recommendations for datatype or language-tags. The latter relaxation means that we would not flag an inconsistency in the following data:

```
# Terminological [http://xmlns.com/foaf/spec/index.rdf]
 foaf:Person owl:disjointWith foaf:Document .
# Assertional [fictional]
 ex:Ted foaf:age 25 .
 ex:Ted foaf:age "25" .
 ex:Ted foaf:gender "male" .
 ex:Ted foaf:gender "Male"@en .
 ex:Ted foaf:birthday "25-05"^^xsd:gMonthDay .
 ex:Ted foaf:birthday "25-05" .
```

With respect to these consistency checking rules, we consider the terminological data to be sound.[35] We again only consider terminological axioms which are authoritatively served by their source (see [25]); for example the following axiom:

`sioc:User owl:disjointWith foaf:Person .`

would have to be served by a document which either `sioc:User` or `foaf:Person` dereferences to (either the FOAF or SIOC vocabulary since the axiom applies to a combination of FOAF and SIOC assertional data).

Given a grounding for such a rule, we wish to analyse the join positions to determine whether or not the inconsistency is caused by consolidation; we are thus only interested in join variables which appear at least once in a consolidatable position (thus, we do not support **dt-not-type**) and where the join variable is "intra-assertional" (exists twice in the assertional patterns).

First note that `owl:sameAs` patterns—particularly in rule **eq-diff1**—are implicit in the consolidated data; e.g., consider:

```
# Assertional [http://www.wikier.org/foaf.rdf]
 wikier:wikier′ owl:differentFrom eswc2006p:sergio-fernandez′ .
```

where an inconsistency is implicitly given by the `owl:sameAs` relation that holds between the consolidated identifiers `wikier:wikier′` and `eswc2006p:sergio-fernandez′`. In this example, there are two Semantic Web researchers, respectively named "Sergio Fernández"[36] and "Sergio Fernández Anzuola"[37] who both participated in the ESWC 2006 conference, and who were subsequently conflated in the "DogFood" export.[38] The former Sergio subsequently added a counter-claim in his FOAF file, asserting the above `owl:differentFrom` statement.

Other inconsistencies do not involve explicit `owl:sameAs` patterns, a subset of which may require "positive" reasoning to be detected; e.g.:

```
# Terminological [http://xmlns.com/foaf/spec]
 foaf:Person owl:disjointWith foaf:Organization .
 foaf:knows rdfs:domain foaf:Person .
# Assertional [http://identi.ca/w3c/foaf]
 identica:48404′ foaf:knows identica:45563 .
# Assertional [inferred by prp-dom]
 identica:48404′ a foaf:Person
# Assertional [http://data.semanticweb.org/organization/w3c/rdf]
 semweborg:w3c′ a foaf:Organization .
```

where the two entities are initially consolidated due to sharing the value `http://www.w3.org/` for the inverse-functional property `foaf:homepage`; the W3C is stated to be a `foaf:Organization` in one document, and is inferred to be a person from its `identi.ca` profile through rule **prp-dom**; finally, the W3C is a member of two disjoint classes, forming an inconsistency detectable by rule **cax-dw**.[39]

In order to resolve inconsistencies, we make three simplifying assumptions:

(i) the steps involved in the consolidation can be rederived with knowledge of direct inlinks and outlinks of the consolidated entity, or reasoned knowledge derived therefrom;
(ii) inconsistencies are caused by pairs of consolidated identifiers;
(iii) we repair individual equivalence classes and do not consider the case where repairing one such class may indirectly repair another.

With respect to the first item, our current implementation will be performing a repair of the equivalence class based on knowledge of direct inlinks and outlinks, available through a simple merge-join as used in the previous section; this thus precludes repair of consolidation found through rule **cls-maxqc2**, which also requires knowledge about the class memberships of the outlinked node. With respect to the second item, we say that inconsistencies are caused by pairs of identifiers—what we term *incompatible identifiers*—such that we

[35] In any case, we always source terminological data from the raw unconsolidated corpus.

[36] `http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/f/Fern=aacute=ndez:Sergio.html`

[37] `http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/a/Anzuola:Sergio_Fern=aacute=ndez.html`

[38] `http://data.semanticweb.org/dumps/conferences/eswc-2006-complete.rdf`

[39] Note that this also could be viewed as a counter-example for using inconsistencies to recant consolidation, where arguably the two entities are coreferent from a practical perspective, even if "incompatible" from a symbolic perspective.

do not consider inconsistencies caused with respect to a single identifier (inconsistencies not caused by consolidation) and do not consider the case where the alignment of more than two identifiers are required to cause a single inconsistency (not possible in our rules) where such a case would again lead to a disjunction of repair strategies. With respect to the third item, it is possible to resolve a set of inconsistent equivalence classes by repairing one; for example, consider rules with multiple "intra-assertional" join-variables (**prp-irp**, **prp-asyp**) which can have explanations involving multiple consolidated identifiers as follows:

```
# Terminological [fictional]
 foaf:made owl:propertyDisjointWith foaf:maker .
# Assertional [fictional]
 ex:AZ" foaf:maker ex:entcons' .
 dblp:Antoine_Zimmermann" foaf:made dblp:HoganZUPD13' .
```

where both equivalences together constitute an inconsistency. Repairing one equivalence class would repair the inconsistency detected for both: we give no special treatment to such a case, and resolve each equivalence class independently—in any case, we find no such incidences in our corpus: these inconsistencies require (i) axioms new in OWL 2 (rules **prp-irp**, **prp-asyp**, **prp-pdw** and **prp-adp**); (ii) alignment of two consolidated sets of identifiers in the subject/object positions. Note that such cases can also occur given the recursive nature of our consolidation—consolidating one set of identifiers may lead to alignments in the join positions of the consolidation rules in the next iteration—however, we did not encounter such recursion during the consolidation phase.

The high-level approach to repairing inconsistent consolidation is as follows:

(i) rederive and build a non-transitive, symmetric graph of equivalences between the identifiers in the equivalence class, based on the inlinks and outlinks of the consolidated entity;
(ii) discover identifiers which together cause inconsistency and must be separated, generating a new seed equivalence class for each, and breaking the direct links between them;
(iii) assign the remaining identifiers into one of the seed equivalence classes based on:
    (a) minimum distance in the non-transitive equivalence class;
    (b) if tied, use a concurrence score.

Given the simplifying assumptions, we can formalise the problem thus: we denote the graph of non-transitive equivalences for a given equivalence class as a weighted graph $G = (V, E, \omega)$ such that $V \subset \mathsf{B} \cup \mathsf{U}$ is the set of vertices, $E \subset \mathsf{B} \cup \mathsf{U} \times \mathsf{B} \cup \mathsf{U}$ is the set of edges, and $\omega : E \mapsto \mathbb{N} \times \mathbb{R}$ is a weighting function for the edges. Our edge weights are pairs (d, c) where d is the number of sets of input triples in the corpus which allow to directly derive the given equivalence relation by means of a direct `owl:sameAs` assertion (in either direction), or a shared inverse-functional object, or functional subject—loosely, the independent evidences for the relation given by the input graph, excluding transitive `owl:sameAs` semantics; c is the concurrence score derivable between the unconsolidated entities and is used to resolve ties (we would expect many strongly connected equivalence graphs where, e.g., the entire equivalence class is given by a single shared value for a given inverse-functional property, and thus require the additional granularity of concurrence for repairing the data in a non-trivial manner). We define a total lexicographical order over these pairs.

Given an equivalence class $Eq \subset \mathsf{U} \cup \mathsf{B}$ which we perceive to cause a *novel* inconsistency—i.e., an inconsistency derivable by the alignment of incompatible identifiers—we first derive a collection of sets $\mathbf{C} = \{C_1, \ldots, C_n\}$, $\mathbf{C} \subset 2^{\mathsf{U} \cup \mathsf{B}}$, such that each $C_i \in \mathbf{C}$, $C_i \subseteq Eq$ denotes an unordered pair of incompatible identifiers.

We then apply a simple *consistent clustering* of the equivalence class, loosely following the notions of a minimal cutting (see, e.g., [46]). For Eq, we create an initial set of singleton sets $\mathbf{Eq}_0$, each containing an individual identifier in the equivalence class. Now let $\Omega(E_i, E_j)$ denote the aggregated weight of the edge considering the merge of the nodes of E_i and the nodes of E_j in the graph: the pair (d, c) such that d denotes the unique evidences for equivalence relations between all nodes in E_i and all nodes in E_j and such that c denotes the concurrence score considering the merge of entities in E_i and E_j—intuitively, the same weight as before, but applied as if the identifiers in E_i and E_j were consolidated in the graph. We can apply the following clustering:

– for each pair of sets $E_i, E_j \in \mathbf{Eq}_n$ such that $\nexists \{a, b\} \in \mathbf{C} : a \in Eq_i, b \in Eq_j$ (i.e., consistently mergeable subsets) identify the weights of $\Omega(E_i, E_j)$ and order the pairings;
– in descending order with respect to the above weights, merge E_i, E_j pairs—such that neither E_i or E_j have already been merged in this iteration—producing E_{n+1} at iteration's end;
– iterate over n until fixpoint.

Thus, we determine the pairs of incompatible identifiers which must necessarily be in different repaired equivalence classes, deconstruct the equivalence class, and then begin reconstructing the repaired equivalence class by iteratively merging the most strongly linked intermediary equivalence classes which will not contain incompatible identifers. [40]

7.2. *Implementing disambiguation*

The implementation of the above disambiguation process can be viewed on two levels: the *macro* level which identifies and collates the information about individual equivalence classes and their respectively consolidated inlinks/outlinks, and the *micro* level which repairs individual equivalence classes.

On the macro level, the task assumes input data sorted by both subject (s, p, o, c, s', o') and object (o, p, s, c, o', s'), again such that s, o represent canonical identifiers and s', o' represent the original identifiers as before. Note that we also require the asserted `owl:sameAs` relations encoded likewise. Given that all the required information about the equivalence classes (their inlinks, outlinks, derivable equivalences and original identifiers) are gathered under the canonical identifiers, we can apply a straight-forward merge-join on s-o over the sorted stream of data and batch consolidated segments of data.

On a micro level, we buffer each individual consolidated segment into an in-memory index; currently, these segments fit in memory, where for the largest equivalence classes we note that inlinks/outlinks are commonly duplicated—if this were not the case, one could consider using an on-disk index which should be feasible given that only small batches of the corpus are under analysis at each given time. We assume access to the relevant terminological knowledge required for reasoning, and the predicate-level statistics derived during from the concurrence analysis. We apply scan-reasoning and inconsistency detection over each batch, and for efficiency, skip over batches which are not symptomised by incompatible identifiers.

For equivalence classes containing incompatible identifiers, we first determine the full set of such pairs through application of the inconsistency detection rules: usually, each detection gives a single pair, where we ignore pairs containing the same identifier (i.e., detections which would equally apply over the unconsolidated data). We check the pairs for a trivial solution: if all identifiers in the equivalence class appear in some pair, we check (i) whether the graph formed by the pairs is strongly connected, in which case, the equivalence class must necessarily be completely disbanded; or (ii) that one identifier is strongly connected to all identifiers in the equivalence class, in which case the strongly connected identifier can be removed from the equivalence class to derive the repair.

For non-trivial repairs, we extract the explicit `owl:sameAs` relations (which we view as directionless) and reinfer `owl:sameAs` relations from the consolidation rules, encoding the subsequent graph. We label edges in the graph with a set of hashes denoting the input triples required for their derivation, such that the cardinality of the hashset corresponds to the primary edge weight. We subsequently use a priority-queue to order the edge-weights, and only materialise concurrency scores in the case of a tie. Nodes in the equivalence graph are merged by combining unique edges and merging the hashsets for overlapping edges. Using these operations, we can apply the aformentioned process to derive the final repaired equivalence classes.

In the final step, we encode the repaired equivalence classes in memory, and perform a final scan of the corpus (in natural sorted order), revising identifiers according to their repaired canonical term.

7.3. *Distributed implementation*

Distribution of the task becomes straight-forward, assuming that the slave machines have knowledge of terminological data, predicate-level statistics, and already have the consolidation encoding sextuples sorted and coordinated by hash on s and o. Note that all of these data are present on the slave machines from previous tasks; for the concurrence analysis, we in fact maintain sextuples during the data preparation phase (although not required by the analysis).

Thus, we are left with two steps:

– **run**: each slave machine performs the above process on it's segment of the corpus, applying a merge-join over the data sorted by $(s, p, o, c, s'o')$ and (o, p, s, c, o', s') to derive batches of consolidated data, which are subsequently analysed, diagnosed, and a repair derived in memory;
– **gather/run**: the master machine gathers all repair information from all slave machines, and floods the merged repairs to the slave machines; the slave machines subsequently perform the final repair of the corpus.

[40] We note the possibility of a dual correspondence between our "bottom-up" approach to repair and the "top-down" minimal hitting set techniques introduced by Reiter [40].

Category	min	% Total
Total execution time	234.8	100
Master (Local)		
Executing	1	0.4
Miscellaneous	1	0.4
Idle (waiting for slaves)	99.6	99.6
Slave (Parallel)		
Avg. Executing (total exc. idle)	205.5	87.5
Identify inconsistencies and repairs	147.8	62.9
Repair Corpus	57.7	24.6
Avg. Idle	29.3	12.5
Waiting for peers	28.3	12.1
Waiting for master	1	0.4

Table 13
Breakdown of timing of distributed disambiguation and repair

7.4. *Performance Evaluation*

The total time taken for inconsistency-based disambiguation was 3.91 h. The inconsistency and equivalence class repair analysis took 2.87 h, with a significant average idle time of 24.4 min (14.16%): in particular, certain large batches of consolidated data took significant amounts of time to process, particularly to reason over.[41] Subsequently repairing the corpus took 1.03 h, with an average idle time of 3.9 min.

In Table 13, we again summarise the timing of the task. Note that the aggregation of the repair information took a negligible amount of time, and where only a total of one minute is spent on the slave machine. Most notably, load-balancing is somewhat of an issue, causing slave machines to be idle for, on average, 12.5% of the total task time, mostly waiting for peers. This percentage—and the general load-balancing characteristic—would likely increase further, given more machines, or a higher scale of data.

7.5. *Results Evaluation*

It seems that our discussion of inconsistency repair has been somewhat academic: from the total of 2.82 million consolidated batches to check, we found 523 equivalence classes (0.019%) causing novel inconsistency. Of these, 23 were detected through `owl:differentFrom` assertions, 94 were detected through distinct literal values for inverse-functional properties, and 406 were detected through disjoint-class constraints. We list the top five functional-properties given non-distinct literal values in Table 14 and the top five disjoint classes in Table 15—note that the `dpo:` functional-properties gave identical detections, and that the class `foaf:Person` is a subclass of `foaf:Agent`, and thus an identical detection is given twice.[42] All equivalence classes were broken into two repaired sub-equivalence class—further, still all had a trivial repair given by separating a single identifier appearing in each incompatible pair (with all original identifiers appearing in some pair). Thus, for the moment, our repair strategy is purely academic.[43]

[41]The additional expense is due to the relaxation of duplicate detection: we cannot consider duplicates on a triple level, but must consider uniqueness based on the entire sextuple to derive the information required for repair. Thus, we must apply many duplicate inferencing steps.

Functional Property	Detections
`foaf:gender`	56
`foaf:age`	32
`dbo:height`/`dbo:height`/`dbo:wheelbase`/`dbo:width`	4
`atomowl:body`	1
`loc:address`	1

Table 14
Breakdown of inconsistency detections for functional-properties, where `dpo:` properties gave identical detections

Disjoint Class 1	Disjoint Class 2	Detections
`foaf:Document`	`foaf:Person`(/`foaf:Agent`)	312
`ecs:Group`	`ecs:Individual`	37
`foaf:Organization`	`foaf:Person`	24
`foaf:Document`	`foaf:Agent`	23
`foaf:Person`	`foaf:Project`	7

Table 15
Breakdown of inconsistency detections for disjoint-classes

8. Critical Discussion

In this section, we provide critical discussion of our approach, following the dimensions of the requirements listed at the outset.

With respect to **scale**, on a high level, our primary means of organising the bulk of the corpus is external-sorts, characterised by the linearithmic time complexity $O(n*\log(n))$; external-sorts do not have a critical main-memory requirement, and are efficiently distributable. Our primary means of accessing the data is via linear scans. With respect to the individual tasks:

– our current baseline consolidation approach relies on an in-memory `owl:sameAs` index: however we

[42]Further, note that between the time of the crawl and the time of writing, the FOAF vocabulary has removed disjointness constraints between the `foaf:Document` and `foaf:Person`/`foaf:Agent` classes.

[43]We also considered a dual form of the concurrence to detect incorrect equivalence classes: for example, to use the quasi-functional nature of `foaf:name` to repair consolidated entities with multiple such values. However, we noted in preliminary results that such analysis gave poor results for our corpus, where we noticed, for example, that (indeed, highly ranked) persons with multiple `foaf:weblog` values—itself measured to be a quasi-functional property—would be identified as incorrect.

demonstrate an on-disk variant in the extended consolidation approach;
– the extended consolidation currently loads terminological data into memory, which is required by all machines: if necessary, we claim that an on-disk terminological index would offer good performance given the distribution of class and property memberships, where we posit that a high cache-hit rate would be enjoyed;
– for the entity concurrency analysis, the predicate level statistics required by all machines is small in volume—for the moment, we do not see this as a serious factor in scaling-up;
– for the inconsistency detection, we identify the same potential issues with respect to terminological data; also, given large equivalence classes with a high number of inlinks and outlinks, we would encounter main-memory problems, where we posit that an on-disk index could be applied assuming a reasonable upper limit on batch sizes.

With respect to **efficiency**:
– the on-disk aggregation of `owl:sameAs` data for the extended consolidation has proven to be a bottleneck—for efficient processing at higher levels of scale, distribution of this task would we a priority, which should be feasible given that again, the primitive operations involved are external sorts and scans, with non-critical in-memory indices to accelerate reaching the fixpoint;
– although we typically observe terminological data to constitute a small percentage of Linked Data corpora (0.1% in our corpus; cf. [23,25]) at higher scales, aggregating the terminological data for all machines may become a bottleneck, and distributed approaches to perform such would need to be investigated; similarly, as we have seen, large terminological documents can cause load-balancing issues; [44]
– for the concurrence analysis and inconsistency detection, data are distributed according to a modulo-hash function on the subject and object position, where we do not hash on the objects of `rdf:type` triples—although we demonstrated even data distribution by this approach for our current corpus, this may not hold in the general case;
– as we have already seen for our corpus and machine count, the complexity of repairing consolidated batches may become an issue given large equivalence class sizes;
– there is some notable idle time for our machines, where the total cost of running the pipeline could be reduced by interleaving jobs.

With the exception of our manually derived blacklist for values of (inverse-)functional-properties, the methods presented herein have been entirely **domain-agnostic** and **fully automatic**.

One major open issue is the question of **precision** and **recall**. Given the nature of the tasks—particularly the scale and diversity of the datasets—we posit that deriving an appropriate gold standard is currently infeasible:
– the scale of the corpus precludes manual or semi-automatic processes;
– any automatic process for deriving the gold standard would make redundant the approach to test;
– results derived from application of the methods on subsets of manually verified data would not be equatable to the results derived from the whole corpus;
– even assuming a manual approach were feasible, oftentimes there is no objective criteria for determining what precisely signifies what—the publisher's original intent is often ambiguous.

Thus, we prefer symbolic approaches to consolidation and disambiguation which are predicated on the formal semantics of the data, where we can appeal to the fact that incorrect consolidation is due to erroneous data, not an erroneous approach. Without a formal means of sufficiently evaluating the results, we employ statistical methods for applications where precision is not a primary requirement. In general, we posit that for the corpora we target, such research can only find it's real litmus test when integrated into a system with a critical user-base.

Finally, we have only briefly discussed issues relating to **web-tolerance**: e.g., spamming or conflicting data. With respect to such consideration, we currently (i) derive and use a blacklist for common void values; (ii) consider authority for terminological data [23,25]; and (iii) try to detect erroneous consolidation through consistency verification. One might question an approach which trusts all equivalences asserted or derived from the data. Along these lines, we track the original pre-consolidation identifiers (in the form of sextuples) which can be used to revert erroneous consolidation. In fact, similar considerations can be applied more generally to the re-use of identifiers across sources: giving special consideration to the consolidation of third party data about an entity is somewhat fallacious without also considering the third party contribution of data using a consistent identifier. In both cases, we track the context

[44] We reduce terminological statements on a document-by-document basis according to unaligned blank-node positions: for example, we prune RDF collections identified by blank-nodes which do not join with, e.g., an `owl:unionOf` axiom.

of (consolidated) statements which at least can be used to verify or post-process sources.[45] Currently, the corpus we evaluate our methods against does not exhibit any significant deliberate spamming, but rather indeliberate noise—we leave more mature means of handling spamming for future work (as required).

9. Related work

Work relating to entity consolidation has been researched in the area of databases for a number of years, aiming to identify and process co-referent signifiers, with works under the titles of record linkage, record fusion, merge-purge, instance fusion, and duplicate identification, and (ironically) a plethora of variations thereupon; see [33,30,7,3,1,2], etc., and a survey at [12]. Unlike our approach, which leverages the declarative semantics of the data in order to be domain agnostic, such systems usually operate given closed schemas—similarly, they typically focus on string-similarity measures and statistical analysis. Haas et al. note that "in relational systems where the data model does not provide primitives for making same-as assertions[...] there is a value-based notion of identity" [18]. However, we note that some works have focussed on leveraging semantics for such tasks in relation databases; e.g., Fan et al. [13] leverage domain knowledge to match entities, where interestingly they state "[r]eal life data is typically dirty... [thus] it is often necessary to hinge on the semantics of the data".

Some other works—moreso related to Information Retrieval and Natural Language Processing—focus on extracting coreferent entity names from unstructured text, tying in moreso with aligning the results of Named Entity Recognition where for example, [44] presents an approach to identify coreferences from a corpus of 3 million natural language "mentions" of persons, where they build compound "entities" out of the individual mentions.

With respect to RDF, one area of research also goes by the name *instance matching*: for example, in 2009, the Ontology Alignment Evaluation Initiative[46] introduced a new test track on instance matching[47].

In [34], the authors present the KnoFuss architecture for aligning data on an assertional level; they identify a three phase process involving coreferencing (finding equivalent individuals), conflict detection (finding inconsistencies caused by the integration), and inconsistency resolution. For the coreferencing, the authors introduce and discuss approaches incorporating string similarity measures and class-based machine learning techniques. Although the high-level process is similar to our own, the authors do not address scalability concerns.

In [42], Scharffe et al. identify four steps in aligning datasets: *align*, *interlink*, *fuse* and *post-process*. The align process identifies equivalences between entities in the two datasets, the interlink process materialises `owl:sameAs` relations between the two datasets, the aligning step merges the two datasets (on both a terminological and assertional level, possibly using domain-specific rules), and the *post-processing* phase subsequently checks the consistency of the output data. Although parts of this process echoes our own, they have yet to demonstrate large-scale evaluation, focussing on datasets containing 2.5 thousand entities.

In [35], the authors present an approach for aligning two A-Boxes described using the same T-Box; in particular they leverage similarity measures introdued in [48], and define an optimisation problem to identify the alignment which generates the most highest weighted similarity between the two A-Boxes under analysis: they use Integer Linear Programming to generate the optimal alignment, encoding linear constraints to enforce *valid* (i.e., consistency preserving), one-to-one, functional mappings. Although they give performance results, they do not directly address scalability. Their method for comparing entities is similar in practice to ours: they measure the "overlapping knowledge" between two entities, counting how many assertions are true about both. The goal is to match entities such that: the resulting consolidation is consistent; the measure of overlap is maximal.

Like us, Castano et al. [6] approach instance matching from two distinct perspectives: (i) determine coreferent identifiers; (ii) to detect similar individuals based on the data they share. Much of their work is similar in principle to ours: in particular, they use reasoning for identifying equivalences and use a statistical approach for identifying properties "with high identification power". They do not consider use of inconsistency detection for disambiguating entities, and perhaps more critically, only evaluate with respect to a dataset containing ∼15 thousand entities.

With respect to URI naming on the Web, Bouquet et al. [5] argue for a centralised naming architecture for minting URI signifiers for the Web; we see such

[45] Although it must be said, we currently do not track the steps used to derive the equivalences involved in consolidation, which would be expensive to materialise and maintain.

[46] OAEI. `http://oaei.ontologymatching.org/`

[47] Instance data matching. `http://www.instancematching.org/`

a centralised "'naming authority" as going against the ad-hoc, decentralised, scale-free nature of the Web.

The Sindice and Sig.ma search systems internally uses inverse-functional properties to find equivalent identifiers [36,50]. Sindice investigates some bespoke "schema-level" reasoning to identify a wider range of inverse-functional properties [36]; however, compared to our approach, they (i) do not use functional properties or cardinality constraints; (ii) would still miss equivalences where identifiers use the same value with different inverse-functional properties, and where, e.g., those properties are in an equivalence or subsumption relationship.

Online systems RKBExplorer [16,15] [48], <sameAs> [49] and ObjectCoref [8] [50] offer on-demand querying for `owl:sameAs` relations found for a given input URI, which they internally compute and store; the former focus on publishing `owl:sameAs` relations for authors and papers in the area of scientific publishing, with the latter two systems offering more general `owl:sameAs` relationships between Linked Data identifiers. In fact, many of the `owl:sameAs` relations we consume are published as Linked Data by the RKBExplorer system.

In [52], the authors present the Silk framework for creating and maintaining inter-linkage between domain-specific RDF datasets; in particular, this framework provides publishers with a means of discovering and creating `owl:sameAs` links between data sources using domain-specific rules and parameters. Thereafter, publishers can integrate discovered links into their exports, enabling better linkage of the data and subsequent consolidation by data consumers: this framework goes hand-in-hand with our approach, producing the `owl:sameAs` relations which we consume.

Popitsch and Haslhofer present discussion on the problem of broken links in Linked Data, identifying structurally broken links (the Web of Data's version of a "deadlink") and semantically broken links, where the original meaning of an identifier changes after a link has been remotely asserted [38]. The authors subsequently present the DSNotify system, which monitors dynamicity over a given subset of Linked Data and can detect and act upon changes—e.g., to notify another agent or correct broken links—and can also be used to indirectly link the dynamic target.

Various authors have looked at applying consolidation over domain-specific RDF corpora: e.g., Sleeman and Finin look at using machine learning techniques to consolidate FOAF personal profile information [45]; Shi et al. similarly look at FOAF-specific alignment techniques [43] using inverse-functional properties and fuzzy string matching; Jentzsch et al. examine alignment of published drug data [27]; [51] Raimond et al. look at interlinking RDF from the music-domain [39]; Monaghan and O' Sullivan apply consolidation to photo annotations expressed in RDF [32].

Salvadores et al. [41] present the LinksB2N system which aims to perform scalable integration of RDF data, particularly focussing on evaluation over corpora from the marketing domain; however, their methods are not specific to this domain. They do not leverage the semantics of the data for performing consolidation, instead using similarity measures, based on the idea that "the unique combination of RDF predicates associated with RDF resources is what defines their existence as unique" [41]. This is a similar intuition to that behind our concurrence analysis, but we again question the validity of such an assumption for consolidation, particularly given incomplete data and the Open World Assumption underlying RDF(S)/OWL—we view an RDF resource as a description of something signified, and would wish to avoid conflating unique signifiers, even if they match *precisely* with respect to their description.

In [19], the authors discuss the semantics and current usage of `owl:sameAs` in Linked Data, discussing issues relating to *identity*, and providing four categories of `owl:sameAs` usage to relate entities which are closely related, but for which the semantics of `owl:sameAs`—particularly substitution—does not quite hold; in fact, this paper serves as a philosophical counterpoint to those aforementioned related works which translate weighted similarities into weighted equivalences. Needless to say, we do not discount such approaches—they may of course of course derive useful and correct alignments not possible through a purely symbolic analysis—but would be cautious when considering using such an approach over arbitrary Linked Data, particularly given the inherent difficulties in evaluating the precision thereof.

In [10], the authors present the idMesh system, which leverages user-defined associations and probabalistic methods to derive entity-level relationships, including resolution of conflicts; they also delineate entities based on "temporal discrimination", whereby coreferent entities may predate or postdate one another, capturing a description thereof at a particular point in time. The

[48] `http://www.rkbexplorer.com/sameAs/`

[49] `http://sameas.org/`

[50] `http://ws.nju.edu.cn/objectcoref/`

[51] In fact, we believe that this work generates the incorrect results observable in Table 2; cf. `http://groups.google.com/group/pedantic-web/browse_thread/thread/ad740f7052cc3a2d`.

idMesh system itself is designed over a peer-to-peer network with centralised coordination. However, evaluation is over synthetic data, where they only demonstrate a maximum scale involving 8,000 entities and 24,000 links, over 400 machines: the evaluation of performance focusses on network traffic and message exchange as opposed to time.

In [28], the authors apply reasoning over 0.9 billion Linked Data triples using the BigOWLIM reasoner; however, this dataset is manually selected as a merge of a number of smaller, known datasets as opposed to an arbitrary corpus. They discuss optimisations similar to our canonicalisation as a necessary means of avoiding the quadratic nature of traditional replacement semantics for `owl:sameAs`.[52]

With respect to distributed consolidation, in [51], Urbani et al. introduced the WebPie system [51] which uses MapReduce to perform pD* reasoning [49] using a cluster of commodity hardware similar to ourselves—the pD* ruleset contains rules for handling `owl:sameAs` replacement, inverse-functional properties and functional-properties, but not for cardinalities (which, in any case we demonstrated to be ineffective over out corpus). The authors also discuss a similar approach to our canonicalisation for handling equivalent identifiers. They demonstrate their methods over 100 billion triples of synthetic LUBM data over 64 machines—however, they do not present evaluation over Linked Data, do not perform any form of similiarity or concurrence measures, do not consider inconsistency detection (not given by the pD* ruleset, or by their corpora) and generally have a somewhat different focus: scalable distributed rule-based materialisation.

10. Conclusion

In this paper, we have provided a comprehensive discussion on scalable and distributed methods for consolidating, matching, and disambiguating entities present in a large static Linked Data corpus. Throughout, we have focussed on the scalability and practicalities of applying our methods over real, arbitrary Linked Data in a domain agnostic and (almost entirely) automatic fashion. We have shown how to use explicit `owl:sameAs` relations in the data to perform consolidation, and subsequently expanded this approach, leveraging the declarative formal semantics of the corpus to materialise additional `owl:sameAs` relations. We also presented a scalable approach to identify weighted entity concurrences: entities which share many inlinks, outlinks, and attribute values—we note that those entities demonstrating the highest concurrence were *not* coreferent. Next, we presented an approach using inconsistencies to disambiguate entities and subsequently repair equivalence classes: we found that this approach currently derives few diagnoses, where the granularity of inconsistencies within Linked Data is not sufficient for accurately pinpointing all incorrect consolidation. Finally, we tempered our contribution with critical discussion, particularly focussing on scalability and efficiency concerns.

We believe that the area of research touched upon in this paper—particularly as applied to large scale Linked Data corpora—is of particular significance given the rapid growth in popularity of Linked Data publishing. As the scale and diversity of the Web of Data expands, scalable and precise data integration technique will become of vital importance, particularly for data warehousing applications—we see the work presented herein as a significant step in the right direction.

[52]We had also presented this in earlier papers [22,23].

References

[1] Riccardo Albertoni and Monica De Martino. Semantic Similarity of Ontology Instances Tailored on the Application Context. In *Proc. of OTM 2006, Part I*, volume 4275 of *LNCS*, pages 1020–1038. Springer, 2006.

[2] Riccardo Albertoni and Monica De Martino. Asymmetric and Context-Dependent Semantic Similarity among Ontology Instances. *Jour. on Data Semantics*, 4900(10):1–30, 2008.

[3] Philip A. Bernstein, Sergey Melnik, and Peter Mork. Interactive Schema Translation with Instance-Level Mappings. In *Proc. of VLDB 2005*, pages 1283–1286. ACM Press, 2005.

[4] Piero A. Bonatti, Aidan Hogan, Axel Polleres, and Luigi Sauro. Robust and Scalable Linked Data Reasoning Incorporating Provenance and Trust Annotations. Available at `http://sw.deri.org/~aidanh/docs/saor_ann_jws_si.pdf`. Under review. Reference to be corrected. Please do not distribute—for review purposes only.

[5] Paolo Bouquet, Heiko Stoermer, Michele Mancioppi, and Daniel Giacomuzzi. OkkaM: Towards a solution to the "identity crisis" on the semantic web. In *Proceedings of SWAP 2006, the 3rd Italian Semantic Web Workshop*, volume 201 of *CEUR Workshop Proceedings*, December 2006.

[6] Silvana Castano, Alfio Ferrara, Stefano Montanelli, and Davide Lorusso. Instance matching for ontology population. In Salvatore Gaglio, Ignazio Infantino, and Domenico Saccà, editors, *SEBD*, pages 121–132, 2008.

[7] Zhaoqi Chen, Dmitri V. Kalashnikov, and Sharad Mehrotra. Exploiting relationships for object consolidation. In *IQIS '05: Proceedings of the 2nd international workshop on Information quality in information systems*, pages 47–58, New York, NY, USA, 2005. ACM Press.

[8] Gong Cheng and Yuzhong Qu. Searching Linked Objects with Falcons: Approach, Implementation and Evaluation. *Int. J. Semantic Web Inf. Syst.*, 5(3):49–70, 2009.

[9] Martine De Cock and Etienne E. Kerre. On (un)suitable fuzzy relations to model approximate equality. *Fuzzy Sets and Systems*, 133(2):137–153, 2003.

[10] Philippe Cudré-Mauroux, Parisa Haghani, Michael Jost, Karl Aberer, and Hermann de Meer. idMesh: Graph-Based Disambiguation of Linked Data. In *WWW*, pages 591–600, 2009.

[11] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI*, pages 137–150, 2004.

[12] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16, 2007.

[13] Wenfei Fan, Xibei Jia, Jianzhong Li, and Shuai Ma. Reasoning about record matching rules. *PVLDB*, 2(1):407–418, 2009.

[14] Javier D. Fernández, Claudio Gutierrez, and Miguel A. Martínez-Prieto. Rdf compression: basic approaches. In *WWW*, pages 1091–1092, 2010.

[15] Hugh Glaser, Afraz Jaffri, and Ian Millard. Managing Co-reference on the Semantic Web. In *Proc. of LDOW 2009*, 2009.

[16] Hugh Glaser, Ian Millard, and Afraz Jaffri. `RKBExplorer.com`: A knowledge driven infrastructure for linked data providers. In *ESWC Demo*, Lecture Notes in Computer Science, pages 797–801. Springer, June 2008.

[17] Bernardo Cuenca Grau, Boris Motik, Zhe Wu, Achille Fokoue, and Carsten Lutz (eds.). OWL 2 Web Ontology Language: Profiles. W3C Working Draft, April 2008. `http://www.w3.org/TR/owl2-profiles/`.

[18] Laura M. Haas, Martin Hentschel, Donald Kossmann, and Renée J. Miller. Schema and data: A holistic approach to mapping, resolution and fusion in information integration. In Alberto H. F. Laender, Silvana Castano, Umeshwar Dayal, Fabio Casati, and José Palazzo Moreira de Oliveira, editors, *ER*, volume 5829 of *Lecture Notes in Computer Science*, pages 27–40. Springer, 2009.

[19] Harry Halpin, Patrick J. Hayes, James P. McCusker, Deborah L. McGuinness, and Henry S. Thompson. When owl:sameAs Isn't the Same: An Analysis of Identity in Linked Data. In *International Semantic Web Conference (1)*, pages 305–320, 2010.

[20] Harry Halpin, Ivan Herman, and Pat Hayes. When owl:sameAs isn't the Same: An Analysis of Identity Links on the Semantic Web. In *Linked Data on the Web WWW2010 Workshop (LDOW2010)*, 2010.

[21] Patrick Hayes. RDF semantics. W3C Recommendation, February 2004. `http://www.w3.org/TR/rdf-mt/`.

[22] Aidan Hogan, Andreas Harth, and Stefan Decker. Performing Object Consolidation on the Semantic Web Data Graph. In *1st I3 Workshop: Identity, Identifiers, Identification Workshop*, 2007.

[23] Aidan Hogan, Andreas Harth, and Axel Polleres. Scalable Authoritative OWL Reasoning for the Web. *Int. J. Semantic Web Inf. Syst.*, 5(2):49–90, 2009.

[24] Aidan Hogan, Andreas Harth, Jürgen Umbrich, Sheila Kinsella, Axel Polleres, and Stefan Decker. Searching and Browsing Linked Data with SWSE: the Semantic Web Search Engine. Technical Report DERI-TR-2010-07-23, Digital Enterprise Research Institute (DERI), 2010. `http://www.deri.ie/fileadmin/documents/DERI-TR-2010-07-23.pdf`.

[25] Aidan Hogan, Jeff Z. Pan, Axel Polleres, and Stefan Decker. SAOR: Template Rule Optimisations for Distributed Reasoning over 1 Billion Linked Data Triples. In *International Semantic Web Conference*, 2010. (to appear).

[26] Aidan Hogan, Axel Polleres, Jürgen Umbrich, and Antoine Zimmermann. Some entities are more equal than others: statistical methods to consolidate Linked Data. In *4th International Workshop on New Forms of Reasoning for the Semantic Web: Scalable and Dynamic (NeFoRS2010)*, 2010.

[27] Anja Jentzsch, Jun Zhao, Oktie Hassanzadeh, Kei-Hoi Cheung, Matthias Samwald, and Bo Andersson. Linking Open Drug Data. In *International Conference on Semantic Systems (I-SEMANTICS?09)*, 2009.

[28] Atanas Kiryakov, Damyan Ognyanoff, Ruslan Velkov, Zdravko Tashev, and Ivan Peikov. LDSR: a Reason-able View to the Web of Linked Data. In *Semantic Web Challenge (ISWC2009)*, 2009.

[29] Frank Klawonn. Should fuzzy equality and similarity satisfy transitivity? comments on the paper by m. de cock and e. kerre. *Fuzzy Sets and Systems*, 133(2):175–180, 2003.

[30] Martin Michalowski, Snehal Thakkar, and Craig A. Knoblock. Exploiting secondary sources for automatic object consolidation. In *Proceeding of 2003 KDD Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, 2003.

[31] B. Scott Michel, Konstantinos Nikoloudakis, Peter L. Reiher, and Lixia Zhang. Url forwarding and compression in adaptive web caching. In *INFOCOM*, pages 670–678, 2000.

[32] Fergal Monaghan and David O'Sullivan. Leveraging ontologies, context and social networks to automate photo annotation. In *SAMT*, pages 252–255, 2007.

[33] H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James. Automatic Linkage of Vital Records: Computers can be used to extract "follow-up" statistics of families from files of routine records. *Science*, 130:954–959, October 1959.

[34] Andriy Nikolov, Victoria S. Uren, Enrico Motta, and Anne N. De Roeck. Integration of semantically annotated data by the knofuss architecture. In *EKAW*, pages 265–274, 2008.

[35] Jan Noessner, Mathias Niepert, Christian Meilicke, and Heiner Stuckenschmidt. Leveraging terminological structure for object reconciliation. In *ESWC (2)*, pages 334–348, 2010.

[36] Eyal Oren, Renaud Delbru, Michele Catasta, Richard Cyganiak, Holger Stenzhorn, and Giovanni Tummarello. Sindice.com: A document-oriented lookup index for open linked data. *Int. J. Metadata Semant. Ontologies*, 3(1):37–52, 2008.

[37] Eyal Oren, Spyros Kotoulas, George Anadiotis, Ronny Siebes, Annette ten Teije, and Frank van Harmelen. Marvin: Distributed reasoning over large-scale Semantic Web data. *J. Web Sem.*, 7(4):305–316, 2009.

[38] Niko Popitsch and Bernhard Haslhofer. Dsnotify: handling broken links in the web of data. In *WWW*, pages 761–770, 2010.

[39] Yves Raimond, Christopher Sutton, and Mark B. Sandler. Interlinking music-related data on the web. *IEEE MultiMedia*, 16(2):52–63, 2009.

[40] Raymond Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, 1987.

[41] Manuel Salvadores, Gianluca Correndo, Bene Rodriguez-Castro, Nicholas Gibbins, John Darlington, and Nigel R. Shadbolt. Linksb2n: Automatic data integration for the semantic web. In *OTM Conferences (2)*, pages 1121–1138, 2009.

[42] F. Scharffe, Y. Liu, and C. Zhou. RDF-AI: an Architecture for RDF Datasets Matching, Fusion and Interlink. In *IJCAI 2009 Workshop on Identity, Reference, and Knowledge Representation (IR-KR)*.

[43] Lian Shi, Diego Berrueta, Sergio Fernández, Luis Polo, and Silvino Fernández. Smushing RDF instances: are Alice and Bob the same open source developer? In *PICKME2008 Workshop*.

[44] Sameer Singh, Michael L. Wick, and Andrew McCallum. Distantly labeling data for large scale cross-document coreference. *CoRR*, abs/1005.4298, 2010.

[45] Jennifer Sleeman and Tim Finin. Learning Co-reference Relations for FOAF Instances. In *Poster and Demo Session at ISWC 2010*.

[46] Mechthild Stoer and Frank Wagner. A simple min-cut algorithm. *J. ACM*, 44(4):585–591, 1997.

[47] Michael Stonebraker. The Case for Shared Nothing. *IEEE Database Eng. Bull.*, 9(1):4–9, 1986.

[48] Heiner Stuckenschmidt. A Semantic Similarity Measure for Ontology-Based Information. In *FQAS '09: Proceedings of the 8th International Conference on Flexible Query Answering Systems*, pages 406–417, Berlin, Heidelberg, 2009. Springer-Verlag.

[49] Herman J. ter Horst. Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. *Journal of Web Semantics*, 3:79–115, 2005.

[50] Giovanni Tummarello, Richard Cyganiak, Michele Catasta, Szymon Danielczyk, and Stefan Decker. Sig.ma: Live views on the Web of Data. In *Semantic Web Challenge (ISWC2009)*, 2009.

[51] Jacopo Urbani, Spyros Kotoulas, Jason Maassen, Frank van Harmelen, and Henri E. Bal. OWL Reasoning with WebPIE: Calculating the Closure of 100 Billion Triples. In *ESWC (1)*, pages 213–227, 2010.

[52] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Discovering and maintaining links on the web of data. In *International Semantic Web Conference*, pages 650–665, 2009.

[53] Denny Vrandečíc, Markus Krötzsch, Sebastian Rudolph, and Uta Lösch. Leveraging non-lexical knowledge for the linked open data web. *Review of April Fool's day Transactions (RAFT)*, 5:18–27, 2010.

[54] Lotfi A. Zadeh, George J. Klir, and Bo Yuan. *Fuzzy Sets, Fuzzy Logic, Fuzzy Systems*. World Scientific Press, 1996.

Appendix A. Prefixes

In Table A.1, we provide the prefixes used throughout the paper.

Prefix	**URI**
	"T-Box prefixes"
atomowl:	`http://bblfish.net/work/atom-owl/2006-06-06/#`
b2r:	`http://bio2rdf.org/bio2rdf:`
b2rr:	`http://bio2rdf.org/bio2rdf_resource:`
dbo:	`http://dbpedia.org/ontology/`
dbp:	`http://dbpedia.org/property/`
ecs:	`http://rdf.ecs.soton.ac.uk/ontology/ecs#`
eurostat:	`http://ontologycentral.com/2009/01/eurostat/ns#`
fb:	`http://rdf.freebase.com/ns/`
foaf:	`http://xmlns.com/foaf/0.1/`
geonames:	`http://www.geonames.org/ontology#`
kwa:	`http://knowledgeweb.../heterogeneity/alignment#`
lldentrezgene:	`http://linkedlifedata.com/resource/entrezgene/`
lldpubmed:	`http://linkedlifedata.com/resource/pubmed/`
loc:	`http://sw.deri.org/2006/07/location/loc#`
mo:	`http://purl.org/ontology/mo/`
mvcb:	`http://webns.net/mvcb/`
opiumfield:	`http://rdf.opiumfield.com/lastfm/spec#`
owl:	`http://www.w3.org/2002/07/owl#`
quaffing:	`http://purl.org/net/schemas/quaffing/`
rdf:	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`
rdfs:	`http://www.w3.org/2000/01/rdf-schema#`
skipinions:	`http://skipforward.net/.../skipinions/`
skos:	`http://www.w3.org/2004/02/skos/core#`
	"A-Box prefixes"
dbpedia:	`http://dbpedia.org/resource/`
eswc2006p:	`http://www.eswc2006.org/people/#`
kingdoms:	`http://lod.geospecies.org/kingdoms/`
macs:	`http://stitch.cs.vu.nl/alignments/macs/`
semweborg:	`http://data.semanticweb.org/organization/`
vperson:	`http://virtuoso.openlinksw.com/dataspace/person/`
wikier:	`http://www.wikier.org/foaf.rdf#`

Table A.1
Used prefixes

Appendix B. OWL 2 RL/RDF rules

OWL2RL	**Antecedent**	**Consequent**
	assertional	
~~eq-ref~~ [a]	~~$?s\ ?p\ ?o$.~~	~~$?s$ owl:sameAs $?s$.~~ ~~$?p$ owl:sameAs $?p$.~~ ~~$?o$ owl:sameAs $?o$.~~
eq-sym	$?x$ owl:sameAs $?y$.	$?y$ owl:sameAs $?x$.
eq-trans	$?x$ owl:sameAs $?y$. $?y$ owl:sameAs $?z$.	$?x$ owl:sameAs $?z$.
eq-rep-s	$?s$ owl:sameAs $?s'$. $?s\ ?p\ ?o$.	$?s'\ ?p\ ?o$.
~~eq-rep-p~~ [b]	~~$?p$ owl:sameAs $?p'$. $?s\ ?p\ ?o$.~~	~~$?s\ ?p'\ ?o$.~~
eq-rep-o [c]	$?o$ owl:sameAs $?o'$. $?s\ ?p\ ?o$.	$?s\ ?p\ ?o'$.

Table B.1
Rules that support the positive semantics of `owl:sameAs`—we use double-strikethrough to denote rules that we do not support by design

[a] We typically omit this rule which adds unnecessary bulk to the materialised inferences, will not lead to any novel consolidation, and could be more easily supported by backward-chaining.
[b] We do not allow `owl:sameAs` inferencing to affect terms in the predicate position of a triple.
[c] We only support this rule for objects of non-`rdf:type` triples.

OWL2RL	Antecedent		Consequent
	terminological	*assertional*	
prp-fp	**?p** a owl:FunctionalProperty .	?x ?p ?y_1 , ?y_2 .	?y_1 owl:sameAs ?y_2 .
prp-ifp	**?p** a owl:InverseFunctionalProperty .	?x_1 ?p ?y . ?x_2 ?p ?y .	?x_1 owl:sameAs ?x_2 .
~~***prp-key***~~	~~**?c** owl:hasKey (**?p_1**, ..., **?p_n**)~~	~~?x ?p_1 ?z_1 ; ... ; ?p_n ?z_n , a ?c .~~ ~~?y ?p_1 ?z_1 ; ... ; ?p_n ?z_n , a ?c .~~	~~?x owl:sameAs ?y .~~
cls-maxc2	**?x** owl:maxCardinality 1 . **?x** owl:onProperty **?p** .	?u a ?x . ?u ?p ?y_1 , ?y_2 .	?y_1 owl:sameAs ?y_2 .
~~***cls-maxqc3***~~	~~**?x** owl:maxQualifiedCardinality 1 .~~ ~~**?x** owl:onProperty **?p** .~~ ~~**?x** owl:onClass **?c** .~~	~~?u a ?x .~~ ~~?u ?p ?y_1 , ?y_2 .~~ ~~?y_1 a ?c . ?y_2 a ?c .~~	~~?y_1 owl:sameAs ?y_2~~
cls-maxqc4	**?x** owl:maxQualifiedCardinality 1 . **?x** owl:onProperty **?p** . **?x** owl:onClass owl:Thing .	?u a ?x . ?u ?p ?y_1 , ?y_2 .	?y_1 owl:sameAs ?y_2

Table B.2
OWL 2 RL/RDF rules that directly produce `owl:sameAs` relations—we denote authoritative variables with bold, we italicise the labels of rules requiring new OWL 2 constructs, and we denote rules not currently supportable by our implementation with strikethrough

OWL2RL	Antecedent	
	terminological	*assertional*
eq-diff1	-	?x owl:sameAs ?y . ?x owl:differentFrom ?y .
~~**eq-diff2**~~	-	~~?x a owl:AllDifferent ;~~ ~~owl:members (?z_1...?z_n) .~~ ~~?z_i owl:sameAs ?z_j . ($i \neq j$)~~
~~***eq-diff3***~~	-	~~?x a owl:AllDifferent ;~~ ~~owl:distinctMembers (?z_1...?z_n) .~~ ~~?z_i owl:sameAs ?z_j . ($i \neq j$)~~
prp-irp	**?p** a owl:IrreflexiveProperty .	?x ?p ?x .
prp-asyp	**?p** a owl:AsymmetricProperty	?x ?p ?y . ?y ?p ?x .
prp-pdw	**?p_1** owl:propertyDisjointWith **?p_2** .	?x ?p_1 ?y ; ?p_2 ?y .
prp-adp	?x a owl:AllDisjointProperties . ?x owl:members (**?p_1**, ..., **?p_n**) .	?u ?p_i ?y ; ?p_j ?y . ($i \neq j$)
~~***prp-npa1***~~	-	~~?x owl:sourceIndividual ?i_1 .~~ ~~?x owl:assertionProperty ?p .~~ ~~?x owl:targetIndividual ?i_2 .~~ ~~?i_1 ?p ?i_2 .~~
~~***prp-npa2***~~	-	~~?x owl:sourceIndividual ?i .~~ ~~?x owl:assertionProperty ?p .~~ ~~?x owl:targetValue ?lt .~~ ~~?i ?p ?lt .~~
~~**cls-nothing2**~~	-	~~?x a owl:Nothing .~~
cls-com	**?c_1** owl:complementOf **?c_2** .	?x a ?c_1 , ?c_2 .
cls-maxc1	**?x** owl:maxCardinality 0 . **?x** owl:onProperty **?p** .	?u a ?x ; ?p ?y .
~~***cls-maxqc1***~~	~~**?x** owl:maxQualifiedCardinality 0 .~~ ~~**?x** owl:onProperty **?p** .~~ ~~**?x** owl:onClass **?c** .~~	~~?u a ?x ; ?p ?y . ?y a ?c .~~
cls-maxqc2	**?x** owl:maxQualifiedCardinality 0 . **?x** owl:onProperty **?p** . **?x** owl:onClass owl:Thing .	?u a ?x ; ?p ?y .
cax-dw	**?c_1** owl:disjointWith **?c_2** .	?x a ?c_1 , ?c_2 .
cax-adc	?x a owl:AllDisjointClasses . ?x owl:members (**?c_1**, ..., **?c_n**) .	?z a ?c_i , ?c_j . ($i \neq j$)
~~**dt-not-type***~~ [a]	-	~~?s ?p ?lt .~~

Table B.4
OWL 2 RL/RDF rules with `false` consequent—we denote authoritative variables with bold, italicise the labels of rules requiring new OWL 2 constructs, denote rules not currently supportable by our implementation with strikethrough, and denote rules not supported by design with double-strikethrough

[a] Where *?lt* is a ill-typed literal: we do not include this rule as it cannot be used to detect incorrect consolidation.

OWL2RL	Antecedent		Consequent
	terminological	*assertional*	
prp-dom	$\boldsymbol{?p}$ rdfs:domain $?c$.	$?x$ $?p$ $?y$.	$?x$ a $?c$.
prp-rng	$\boldsymbol{?p}$ rdfs:range $?c$.	$?x$ $?p$ $?y$.	$?y$ a $?c$.
prp-symp	$\boldsymbol{?p}$ a owl:SymmetricProperty .	$?x$ $?p$ $?y$.	$?y$ $?p$ $?x$.
prp-spo1	$\boldsymbol{?p_1}$ rdfs:subPropertyOf $?p_2$.	$?x$ $?p_1$ $?y$.	$?x$ $?p_2$ $?y$.
prp-eqp1	$\boldsymbol{?p_1}$ owl:equivalentProperty $?p_2$.	$?x$ $?p_1$ $?y$.	$?x$ $?p_2$ $?y$.
prp-eqp2	$?p_1$ owl:equivalentProperty $\boldsymbol{?p_2}$.	$?x$ $?p_2$ $?y$.	$?x$ $?p_1$ $?y$.
prp-inv1	$\boldsymbol{?p_1}$ owl:inverseOf $?p_2$.	$?x$ $?p_1$ $?y$.	$?y$ $?p_2$ $?x$.
prp-inv2	$?p_1$ owl:inverseOf $\boldsymbol{?p_2}$.	$?x$ $?p_2$ $?y$.	$?y$ $?p_1$ $?x$.
cls-int2	$\boldsymbol{?c}$ owl:intersectionOf ($?c_1$ ... $?c_n$) .	$?x$ a $?c$.	$?x$ a $?c_1 ... ?c_n$.
cls-uni	$?c$ owl:unionOf ($?c_1 ... \boldsymbol{?c_i} ... ?c_n$) .	$?x$ a $?c_i$	$?x$ a $?c$.
cls-svf2	$?x$ owl:someValuesFrom owl:Thing ; owl:onProperty $\boldsymbol{?p}$.	$?u$ $?p$ $?v$.	$?u$ a $?x$.
cls-hv1	$\boldsymbol{?x}$ owl:hasValue $?y$; owl:onProperty $?p$.	$?u$ a $?x$.	$?u$ $?p$ $?y$.
cls-hv2	$?x$ owl:hasValue $\boldsymbol{?y}$; owl:onProperty $\boldsymbol{?p}$.	$?u$ $?p$ $?y$.	$?u$ a $?x$.
cax-sco	$\boldsymbol{?c_1}$ rdfs:subClassOf $?c_2$.	$?x$ a $?c_1$.	$?x$ a $?c_2$.
cax-eqc1	$\boldsymbol{?c_1}$ owl:equivalentClass $?c_2$.	$?x$ a $?c_1$.	$?x$ a $?c_2$.
cax-eqc2	$?c_1$ owl:equivalentClass $\boldsymbol{?c_2}$.	$?x$ a $?c_2$.	$?x$ a $?c_1$.

Table B.3
OWL 2 RL/RDF rules containing precisely one assertional pattern in the body, with authoritative variables in bold (see [25])